Network Working Group
Request for Comments: 3303
Category: Informational

P. Srisuresh
Kuokoa Networks
J. Kuthan
Fraunhofer Institute FOKUS
J. Rosenberg
dynamicsoft
A. Molitor
Aravox Technologies
A. Rayhan
Ryerson University

August 2002

# Middlebox communication architecture and framework

## Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

## Copyright Notice



## Abstract

A principal objective of this document is to describe the underlying framework of middlebox communications (MIDCOM) to enable complex applications through the middleboxes, seamlessly using a trusted third party. This document and a companion document on MIDCOM requirements ([REQMTS]) have been created as a precursor to rechartering the MIDCOM working group.

There are a variety of intermediate devices in the Internet today that require application intelligence for their operation. Datagrams pertaining to real-time streaming applications, such as SIP and H.323, and peer-to-peer applications, such as Napster and NetMeeting, cannot be identified by merely examining packet headers. Middleboxes implementing Firewall and Network Address Translator services typically embed application intelligence within the device for their operation. The document specifies an architecture and framework in which trusted third parties can be delegated to assist the middleboxes to perform their operation, without resorting to embedding application intelligence. Doing this will allow a middlebox to continue to provide the services, while keeping the middlebox application agnostic.

## 1. Introduction

Intermediate devices requiring application intelligence are the subject of this document. These devices are referred to as middleboxes throughout the document. Many of these devices enforce application specific policy based functions such as packet filtering, VPN (Virtual Private Network) tunneling, Intrusion detection, security and so forth. Network Address Translator service, on the other hand, provides routing transparency across address realms (within IPv4 routing network or across V4 and V6 routing realms), independent of applications. Application Level Gateways (ALGs) are used in conjunction with NAT to examine and optionally modify application payload so the end-to-end application behavior remains unchanged for many of the applications traversing NAT middleboxes. There may be other types of services requiring embedding application intelligence in middleboxes for their operation. The discussion scope of this document is however limited to Firewall and NAT services. Nonetheless, the MIDCOM framework is designed to be extensible to support the deployment of new services.

Tight coupling of application intelligence with middleboxes makes maintenance of middleboxes hard with the advent of new applications. Built-in application awareness typically requires updates of operating systems with new applications or newer versions of existing applications. Operators requiring support for newer applications will not be able to use third party software/hardware specific to the application and are at the mercy of their middlebox vendor to make the necessary upgrade. Further, embedding intelligence for a large number of application protocols within the same middlebox increases complexity of the middlebox and is likely to be error prone and degrade in performance.

This document describes a framework in which application intelligence can be moved from middleboxes into external MIDCOM agents. The premise of the framework is to devise a MIDCOM protocol that is application independent, so the middleboxes can stay focused on services such as firewall and NAT. The framework document includes some explicit and implied requirements for the MIDCOM protocol. However, it must be noted that these requirements are only a subset. A separate requirements document lists the requirements in detail.

MIDCOM agents with application intelligence can assist the middleboxes through the MIDCOM protocol in permitting applications such as FTP, SIP and H.323. The communication between a MIDCOM agent and a middlebox will not be noticeable to the end-hosts that take part in the application, unless one of the end-hosts assumes the role of a MIDCOM agent. Discovery of middleboxes or MIDCOM agents in the

path of an application instance is outside the scope of this document. Further, any communication amongst middleboxes is also outside the scope of this document.

This document describes the framework in which middlebox communication takes place and the various elements that constitute the framework. Section 2 describes the terms used in the document. Section 3 defines the architectural framework of a middlebox for communication with MIDCOM agents. The remaining sections cover the components of the framework, illustration using sample flows, and operational considerations with the MIDCOM architecture. Section 4 describes the nature of MIDCOM protocol. Section 5 identifies entities that could potentially host the MIDCOM agent function. Section 6 considers the role of Policy server and its function with regard to communicating MIDCOM agent authorization policies. Section 7 is an illustration of SIP flows using a MIDCOM framework in which the MIDCOM agent is co-resident on a SIP proxy server. Section 8 addresses operational considerations in deploying a protocol adhering to the framework described here. Section 9 is an applicability statement, scoping the location of middleboxes. Section 11 outlines security considerations for the middlebox in view of the MIDCOM framework.

## 2. Terminology

Below are the definitions for the terms used throughout the document.

### 2.1. Middlebox function/service

A middlebox function or a middlebox service is an operation or method performed by a network intermediary that may require application specific intelligence for its operation. Policy based packet filtering (a.k.a. firewall), Network address translation (NAT), Intrusion detection, Load balancing, Policy based tunneling and IPsec security are all examples of a middlebox function (or service).

### 2.2. Middlebox

A Middlebox is a network intermediate device that implements one or more of the middlebox services. A NAT middlebox is a middlebox implementing NAT service. A firewall middlebox is a middlebox implementing firewall service.

Traditional middleboxes embed application intelligence within the device to support specific application traversal. Middleboxes supporting the MIDCOM protocol will be able to externalize application intelligence into MIDCOM agents. In reality, some of the

middleboxes may continue to embed application intelligence for certain applications and depend on MIDCOM protocol and MIDCOM agents for the support of remaining applications.

## 2.3. Firewall

Firewall is a policy based packet filtering middlebox function, typically used for restricting access to/from specific devices and applications. The policies are often termed Access Control Lists (ACLs).

## 2.4. NAT

Network Address Translation is a method by which IP addresses are mapped from one address realm to another, providing transparent routing to end-hosts. Transparent routing here refers to modifying end-node addresses en-route and maintaining state for these updates so that when a datagram leaves one realm and enters another, datagrams pertaining to a session are forwarded to the right end-host in either realm. Refer to [NAT-TERM] for the definition of Transparent routing, various NAT types, and the associated terms in use. Two types of NAT are most common. Basic-NAT, where only an IP address (and the related IP, TCP/UDP checksums) of packets is altered and NAPT (Network Address Port Translation), where both an IP address and a transport layer identifier, such as a TCP/UDP port (and the related IP, TCP/UDP checksums), are altered.

The term NAT in this document is very similar to the IPv4 NAT described in [NAT-TERM], but is extended beyond IPv4 networks to include the IPv4-v6 NAT-PT described in [NAT-PT]. While the IPv4 NAT [NAT-TERM] translates one IPv4 address into another IPv4 address to provide routing between private V4 and external V4 address realms, IPv4-v6 NAT-PT [NAT-PT] translates an IPv4 address into an IPv6 address, and vice versa, to provide routing between a V6 address realm and an external V4 address realm.

Unless specified otherwise, NAT in this document is a middlebox function referring to both IPv4 NAT, as well as IPv4-v6 NAT-PT.

## 2.5. Proxy

A proxy is an intermediate relay agent between clients and servers of an application, relaying application messages between the two. Proxies use special protocol mechanisms to communicate with proxy clients and relay client data to servers and vice versa. A Proxy terminates sessions with both the client and the server, acting as server to the end-host client and as client to the end-host server.

Applications such as FTP, SIP, and RTSP use a control session to establish data sessions. These control and data sessions can take divergent paths. While a proxy can intercept both the control and data sessions, it might intercept only the control session. This is often the case with real-time streaming applications such as SIP and RTSP.

## 2.6. ALG

Application Level Gateways (ALGs) are entities that possess the application specific intelligence and knowledge of an associated middlebox function. An ALG examines application traffic in transit and assists the middlebox in carrying out its function.

An ALG may be a co-resident with a middlebox or reside externally, communicating through a middlebox communication protocol. It interacts with a middlebox to set up state, access control filters, use middlebox state information, modify application specific payload, or perform whatever else is necessary to enable the application to run through the middlebox.

ALGs are different from proxies. ALGs are not visible to end-hosts, unlike the proxies which are relay agents terminating sessions with both end-hosts. ALGs do not terminate sessions with either end-host. Instead, ALGs examine, and optionally modify, application payload content to facilitate the flow of application traffic through a middlebox. ALGs are middlebox centric, in that they assist the middleboxes in carrying out their function, whereas, the proxies act as a focal point for application servers, relaying traffic between application clients and servers.

ALGs are similar to Proxies, in that, both ALGs and proxies facilitate Application specific communication between clients and servers.

## 2.7. End-Hosts

End-hosts are entities that are party to a networked application instance. End-hosts referred to in this document, are specifically those terminating Real-time streaming Voice-over-IP applications, such as SIP and H.323, and peer-to-peer applications such as Napster and NetMeeting.

## 2.8. MIDCOM Agents

MIDCOM agents are entities performing ALG functions, logically external to a middlebox. MIDCOM agents possess a combination of application awareness and knowledge of the middlebox function. This

combination enables the agents to facilitate traversal of the middlebox by the application's packets. A MIDCOM agent may interact with one or more middleboxes.

Only "In-Path MIDCOM agents" are considered in this document. In-Path MIDCOM agents are agents which are within the path of those datagrams that the agent needs to examine and/or modify in fulfilling its role as a MIDCOM agent. "Within the path" here simply means that the packets in question flow through the node that hosts the agent. The packets may be addressed to the agent node at the IP layer. Alternatively they may not be addressed to the agent node, but may be constrained by other factors to flow through it. In fact, it is immaterial to the MIDCOM protocol which of these is the case. Some examples of In-Path MIDCOM agents are application proxies, gateways, or even end-hosts that are party to the application.

Agents not resident on nodes that are within the path of their relevant application flows are referred to as "Out-of-Path (OOP) MIDCOM agents" and are out of the scope of this document.

## 2.9. MIDCOM PDP

MIDCOM Policy Decision Point (PDP) is primarily a Policy Decision Point(PDP), as defined in [POL-TERM]; and also acts as a policy repository, holding MIDCOM related policy profiles in order to make authorization decisions. [POL-TERM] defines a PDP as "a logical entity that makes policy decisions for itself or for other network elements that request such decisions"; and a policy repository as "a specific data store that holds policy rules, their conditions and actions, and related policy data".

A middlebox and a MIDCOM PDP may communicate further if the MIDCOM PDP's policy changes or if a middlebox needs further information. The MIDCOM PDP may, at anytime, notify the middlebox to terminate authorization for an agent.

The protocol facilitating the communication between a middlebox and MIDCOM PDP need not be part of the MIDCOM protocol. Section 6 in the document addresses the MIDCOM PDP interface and protocol framework independent of the MIDCOM framework.

Application specific policy data and policy interface between an agent or application endpoint and a MIDCOM PDP is out of bounds for this document. The MIDCOM PDP issues addressed in the document are focused at an aggregate domain level as befitting the middlebox. For example, a SIP MIDCOM agent may choose to query a MIDCOM PDP for the administrative (or corporate) domain to find whether a certain user is allowed to make an outgoing call. This type of application

specific policy data, as befitting an end user, is out of bounds for the MIDCOM PDP considered in this document. It is within bounds, however, for the MIDCOM PDP to specify the specific end-user applications (or tuples) for which an agent is permitted to be an ALG.

### 2.10. Middlebox Communication (MIDCOM) protocol

The protocol between a MIDCOM agent and a middlebox allows the MIDCOM agent to invoke services of the middlebox and allow the middlebox to delegate application specific processing to the MIDCOM agent. The MIDCOM protocol allows the middlebox to perform its operation with the aid of MIDCOM agents, without resorting to embedding application intelligence. The principal motivation behind architecting this protocol is to enable complex applications through middleboxes, seamlessly using a trusted third party, i.e., a MIDCOM agent.

This is a protocol yet to be devised.

### 2.11. MIDCOM agent registration

A MIDCOM agent registration is defined as the process of provisioning agent profile information with the middlebox or a MIDCOM PDP. MIDCOM agent registration is often a manual operation performed by an operator rather than the agent itself.

A MIDCOM agent profile may include agent authorization policy (i.e., session tuples for which the agent is authorized to act as ALG), agent-hosting-entity (e.g., Proxy, Gateway, or end-host which hosts the agent), agent accessibility profile (including any host level authentication information), and security profile (for the messages exchanged between the middlebox and the agent).

### 2.12. MIDCOM session

A MIDCOM session is defined to be a lasting association between a MIDCOM agent and a middlebox. The MIDCOM session is not assumed to imply any specific transport layer protocol. Specifically, this should not be construed as referring to a connection-oriented TCP protocol.

### 2.13. Filter

A filter is packet matching information that identifies a set of packets to be treated a certain way by a middlebox. This definition is consistent with [POL-TERM], which defines a filter as "A set of

terms and/or criteria used for the purpose of separating or categorizing. This is accomplished via single- or multi-field matching of traffic header and/or payload data".

5-Tuple specification of packets in the case of a firewall and 5-tuple specification of a session in the case of a NAT middlebox function are examples of a filter.

### 2.14. Policy action (or) Action

Policy action (or Action) is a description of the middlebox treatment/service to be applied to a set of packets. This definition is consistent with [POL-TERM], which defines a policy action as "Definition of what is to be done to enforce a policy rule, when the conditions of the rule are met. Policy actions may result in the execution of one or more operations to affect and/or configure network traffic and network resources".

NAT Address-BIND (or Port-BIND in the case of NAPT) and firewall permit/deny action are examples of an Action.

### 2.15. Policy rule(s)

The combination of one or more filters and one or more actions. Packets matching a filter are to be treated as specified by the associated action(s). The Policy rules may also contain auxiliary attributes such as individual rule type, timeout values, creating agent, etc.

Policy rules are communicated through the MIDCOM protocol.

## 3.0 Architectural framework for middleboxes

A middlebox may implement one or more of the middlebox functions selectively on multiple interfaces of the device. There can be a variety of MIDCOM agents interfacing with the middlebox to communicate with one or more of the middlebox functions on an interface. As such, the middlebox communication protocol must allow for selective communication between a specific MIDCOM agent and one or more middlebox functions on the interface. The following diagram identifies a possible layering of the service supported by a middlebox and a list of MIDCOM agents that might interact with it.

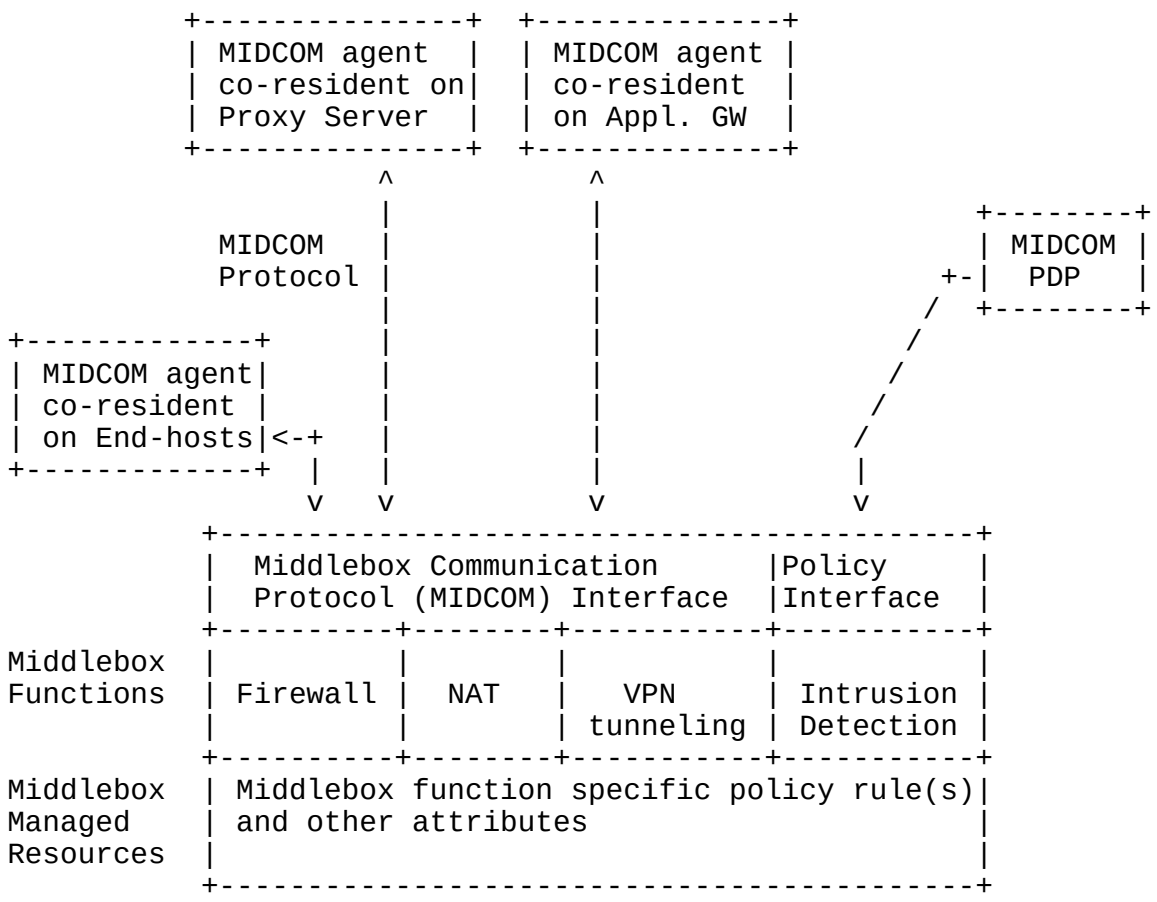



Figure 1: MIDCOM agents interfacing with a middlebox

Firewall ACLs, NAT-BINDs, NAT address-maps and Session-state are a few of the middlebox function specific policy rules. A session state may include middlebox function specific attributes, such as timeout values, NAT translation parameters (i.e., NAT-BINDS), and so forth. As Session-state may be shared across middlebox functions, a Session-state may be created by a function, and terminated by a different function. For example, a session-state may be created by the firewall function, but terminated by the NAT function, when a session timer expires.

Application specific MIDCOM agents (co-resident on the middlebox or external to the middlebox) would examine the IP datagrams and help identify the application the datagram belongs to, and assist the middlebox in performing functions unique to the application and the middlebox service. For example, a MIDCOM agent, assisting a NAT middlebox, might perform payload translations, whereas a MIDCOM agent

assisting a firewall middlebox might request the firewall to permit access to application specific, dynamically generated, session traffic.

## 4. MIDCOM Protocol

The MIDCOM protocol between a MIDCOM agent and a middlebox allows the MIDCOM agent to invoke services of the middlebox and allow the middlebox to delegate application specific processing to the MIDCOM agent. The protocol will allow MIDCOM agents to signal the middleboxes, to let complex applications using dynamic port based sessions through them (i.e., middleboxes) seamlessly.

It is important to note that an agent and a middlebox can be on the same physical device. In such a case, they may communicate using a MIDCOM protocol message formats, but using a non-IP based transport, such as IPC messaging (or) they may communicate using well-defined API/DLL (or) the application intelligence is fully embedded into the middlebox service (as it is done today in many stateful inspection firewall devices and NAT devices).

The MIDCOM protocol will consist of a session setup phase, run-time session phase, and a session termination phase.

Session setup must be preceded by registration of the MIDCOM agent with either the middlebox or the MIDCOM PDP. The MIDCOM agent access and authorization profile may either be pre-configured on the middlebox (or) listed on a MIDCOM PDP; the middlebox is configured to consult. MIDCOM shall be a client-server protocol, initiated by the agent.

A MIDCOM session may be terminated by either of the parties. A MIDCOM session termination may also be triggered by (a) the middlebox or the agent going out of service and not being available for further MIDCOM operations, or (b) the MIDCOM PDP notifying the middlebox that a particular MIDCOM agent is no longer authorized.

The MIDCOM protocol data exchanged during run-time is governed principally by the middlebox services the protocol supports. Firewall and NAT middlebox services are considered in this document. Nonetheless, the MIDCOM framework is designed to be extensible to support the deployment of other services as well.

## 5.0. MIDCOM Agents

MIDCOM agents are logical entities which may reside physically on nodes external to a middlebox, possessing a combination of application awareness and knowledge of middlebox function. A MIDCOM agent may communicate with one or more middleboxes. The issues of middleboxes discovering agents, or vice versa, are outside the scope of this document. The focus of the document is the framework in which a MIDCOM agent communicates with a middlebox using MIDCOM protocol, which is yet to be devised. Specifically, the focus is restricted to just the In-Path agents.

In-Path MIDCOM agents are MIDCOM agents that are located naturally within the message path of the application(s) they are associated with. Bundled session applications, such as H.323, SIP, and RTSP which have separate control and data sessions, may have their sessions take divergent paths. In those scenarios, In-Path MIDCOM agents are those that find themselves in the control path. In a majority of cases, a middlebox will likely require the assistance of a single agent for an application in the control path alone. However, it is possible that a middlebox function, or a specific application traversing the middlebox might require the intervention of more than a single MIDCOM agent for the same application, one for each sub-session of the application.

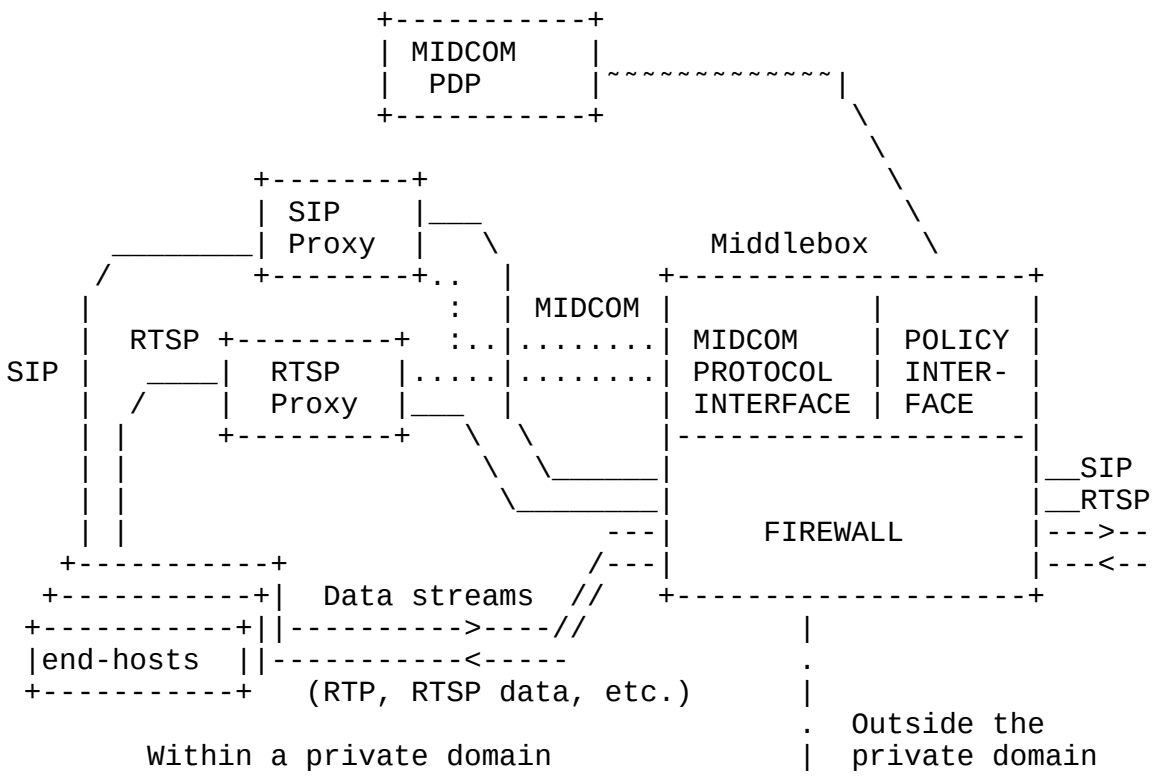
Application Proxies and gateways are a good choice for In-Path MIDCOM agents, as these entities by definition, are in the path of an application between a client and server. In addition to hosting the MIDCOM agent function, these natively in-path application specific entities may also enforce application-specific choices locally, such as dropping messages infected with known viruses, or lacking user authentication. These entities can be interjecting both the control and data sessions. For example, FTP control and Data sessions are interjected by an FTP proxy server.

However, proxies may also be interjecting just the control session and not the data sessions, as is the case with real-time streaming applications, such as SIP and RTSP. Note, applications may not always traverse a proxy and some applications may not have a proxy server available.

SIP proxies and H.323 gatekeepers may be used to host MIDCOM agent functions to control middleboxes implementing firewall and NAT functions. The advantage of using in-path entities, as opposed to creating an entirely new agent, is that the in-path entities already possess application intelligence. You will need to merely enable the use of the MIDCOM protocol to be an effective MIDCOM agent. Figure 2 below illustrates a scenario where the in-path MIDCOM agents

interface with the middlebox. Let us say, the MIDCOM PDP has pre-configured the in-path proxies as trusted MIDCOM agents on the middlebox and the packet filter implements a 'default-deny' packet filtering policy. Proxies use their application-awareness knowledge to control the firewall function and selectively permit a certain number of voice stream sessions dynamically using MIDCOM protocol.

In the illustration below, the proxies and the MIDCOM PDP are shown inside a private domain. The intent however, is not to imply that they be inside the private boundary alone. The proxies may also reside external to the domain. The only requirement is that there be a trust relationship with the middlebox.




Legend: ---- Application data path datagrams
____ Application control path datagrams
.... Middlebox Communication Protocol (MIDCOM)
~~~~ MIDCOM PDP Interface
| private domain Boundary

Figure 2: In-Path MIDCOM Agents for middlebox Communication
~~~~

### 5.1. End-hosts as In-Path MIDCOM agents

End-hosts are another variation of In-Path MIDCOM agents. Unlike Proxies, End-hosts are a direct party to the application and possess all the end-to-end application intelligence there is to it. End-hosts presumably terminate both the control and data paths of an application. Unlike other entities hosting MIDCOM agents, end-host is able to process secure datagrams. However, the problem would be one of manageability - upgrading all the end-hosts running a specific application.

## 6.0. MIDCOM PDP functions

The functional decomposition of the MIDCOM architecture assumes the existence of a logical entity, known as MIDCOM PDP, responsible for performing authorization and related provisioning services for the middlebox as depicted in figure 1. The MIDCOM PDP is a logical entity which may reside physically on a middlebox or on a node external to the middlebox. The protocol employed for communication between the middlebox and the MIDCOM PDP is unrelated to the MIDCOM protocol.

Agents are registered with a MIDCOM PDP for authorization to invoke services of the middlebox. The MIDCOM PDP maintains a list of agents that are authorized to connect to each of the middleboxes the MIDCOM PDP supports. In the context of the MIDCOM Framework, the MIDCOM PDP does not assist a middlebox in the implementation of the services it provides.

The MIDCOM PDP acts in an advisory capacity to a middlebox, to authorize or terminate authorization for an agent attempting connectivity to the middlebox. The primary objective of a MIDCOM PDP is to communicate agent authorization information, so as to ensure that the security and integrity of a middlebox is not jeopardized. Specifically, the MIDCOM PDP should associate a trust level with each agent attempting to connect to a middlebox and provide a security profile. The MIDCOM PDP should be capable of addressing cases when end-hosts are agents to the middlebox.

### 6.1. Authentication, Integrity and Confidentiality

Host authenticity and individual message security are two distinct types of security considerations. Host authentication refers to credentials required of a MIDCOM agent to authenticate itself to the middlebox and vice versa. When authentication fails, the middlebox must not process signaling requests received from the agent that failed authentication. Two-way authentication should be supported. In some cases, the 2-way authentication may be tightly linked to the

establishment of keys to protect subsequent traffic. Two-way authentication is often required to prevent various active attacks on the MIDCOM protocol and secure establishment of keying material.

Security services such as authentication, data integrity, confidentiality and replay protection may be adapted to secure MIDCOM messages in an untrusted domain. Message authentication is the same as data origin authentication and is an affirmation that the sender of the message is who it claims to be. Data integrity refers to the ability to ensure that a message has not been accidentally, maliciously or otherwise altered or destroyed. Confidentiality is the encryption of a message with a key, so that only those in possession of the key can decipher the message content. Lastly, replay protection is a form of sequence integrity, so when an intruder plays back a previously recorded sequence of messages, the receiver of the replay messages will simply drop the replay messages into bit-bucket. Certain applications of the MIDCOM protocol might require support for non-repudiation as an option of the data integrity service. Typically, support for non-repudiation is required for billing, service level agreements, payment orders, and receipts for delivery of service.

IPsec AH ([IPSEC-AH]) offers data-origin authentication, data integrity and protection from message replay. IPsec ESP ([IPSEC-ESP]) provides data-origin authentication to a lesser degree (same as IPsec AH if the MIDCOM transport protocol turns out to be TCP or UDP), message confidentiality, data integrity and protection from replay. Besides the IPsec based protocols, there are other security options as well. TLS based transport layer security is one option. There are also many application-layer security mechanisms available. Simple Source-address based security is a minimal form of security and should be relied on only in the most trusted environments, where those hosts will not be spoofed.

The MIDCOM message security shall use existing standards, whenever the existing standards satisfy the requirements. Security shall be specified to minimize the impact on sessions that do not use the security option. Security should be designed to avoid introducing and to minimize the impact of denial of service attacks. Some security mechanisms and algorithms require substantial processing or storage, in which case the security protocols should protect themselves as well as against possible flooding attacks that overwhelm the endpoint (i.e., the middlebox or the agent) with such processing. For connection oriented protocols (such as TCP) using security services, the security protocol should detect premature closure or truncation attacks.

### 6.2. Registration and deregistration of MIDCOM agents

Prior to allowing MIDCOM agents to invoke services of the middlebox, a registration process must take place. Registration is a different process than establishing a MIDCOM session. The former requires provisioning agent profile information with the middlebox or a MIDCOM PDP. Agent registration is often a manual operation performed by an operator rather than the agent itself. Setting up MIDCOM session refers to establishing a MIDCOM transport session and exchanging security credentials between an agent and a middlebox. The transport session uses the registered information for session establishment.

Profile of a MIDCOM agent includes agent authorization policy (i.e., session tuples for which the agent is authorized to act as ALG), agent-hosting-entity (e.g., Proxy, Gateway or end-host which hosts the agent), agent accessibility profile (including any host level authentication information) and security profile (i.e., security requirements for messages exchanged between the middlebox and the agent).

MIDCOM agent profile may be pre-configured on a middlebox. Subsequent to that, the agent may choose to initiate a MIDCOM session prior to any data traffic. For example, MIDCOM agent authorization policy for a middlebox service may be preconfigured by specifying the agent in conjunction with a filter. In the case of a firewall, for example, the ACL tuple may be altered to reflect the optional Agent presence. The revised ACL may look something like the following.

(<Session-Direction>, <Source-Address>, <Destination-Address>, <IP-Protocol>, <Source-Port>, <Destination-Port>, <Agent>)

The reader should note that this is an illustrative example and not necessarily the actual definition of an ACL tuple. The formal description of the ACL is yet to be devised. Agent accessibility information should also be provisioned. For a MIDCOM agent, accessibility information includes the IP address, trust level, host authentication parameters and message authentication parameters. Once a session is established between a middlebox and a MIDCOM agent, that session should be usable with multiple instances of the application(s), as appropriate. Note, all of this could be captured in an agent profile for ease of management.

The technique described above is necessary for the pre-registration of MIDCOM agents with the middlebox. The middlebox provisioning may remain unchanged, if the middlebox learns of the registered agents through a MIDCOM PDP. In either case, the MIDCOM agent should initiate the session prior to the start of the application. If the agent session is delayed until after the application has started, the

agent might be unable to process the control stream to permit the data sessions. When a middlebox notices an incoming MIDCOM session, and the middlebox has no prior profile of the MIDCOM agent, the middlebox will consult its MIDCOM PDP for authenticity, authorization, and trust guidelines for the session.

## 7.0. MIDCOM Framework Illustration using an In-Path agent

In figure 3 below, we consider SIP applications (Refer [SIP]) to illustrate the operation of the MIDCOM protocol. Specifically, the application assumes that a caller, external to a private domain, initiates the call. The middlebox is assumed to be located at the edge of the private domain. A SIP phone (SIP User Agent Client/Server) inside the private domain is capable of receiving calls from external SIP phones. The caller uses a SIP Proxy, node located external to the private domain, as its outbound proxy. No interior proxy is assumed for the callee. Lastly, the external SIP proxy node is designated to host the MIDCOM agent function.

Arrows 1 and 8 in the figure below refer to a SIP call setup exchange between the external SIP phone and the SIP proxy. Arrows 4 and 5 refer to a SIP call setup exchange between the SIP proxy and the interior SIP phone, and are assumed to be traversing the middlebox. Arrows 2, 3, 6 and 7 below, between the SIP proxy and the middlebox, refer to MIDCOM communication. Na and Nb represent RTP/RTCP media traffic (Refer [RTP]) path in the external network. Nc and Nd represent media traffic inside the private domain.

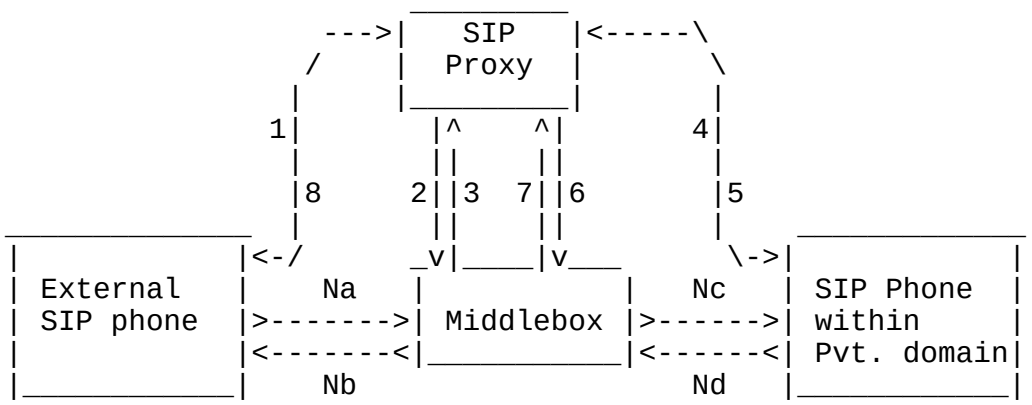



Figure 3: MIDCOM framework illustration with In-Path SIP Proxy

As for the SIP application, we make the assumption that the middlebox is pre-configured to accept SIP calls into the private SIP phone. Specifically, this would imply that the middlebox implementing firewall service is pre-configured to permit SIP calls (destination

TCP or UDP port number set to 5060) into the private phone. Likewise, middlebox implementing NAPT service would have been pre-configured to provide a port binding, to permit incoming SIP calls to be redirected to the specific private SIP phone. I.e., the INVITE from the external caller is not made to the private IP address, but to the NAPT external address.

The objective of the MIDCOM agent in the following illustration is to merely permit the RTP/RTCP media stream (Refer [RTP]) through the middlebox, when using the MIDCOM protocol architecture outlined in the document. A SIP session typically establishes two RTP/RTCP media streams - one from the callee to the caller and another from the caller to the callee. These media sessions are UDP based and will use dynamic ports. The dynamic ports used for the media stream are specified in the SDP section (Refer [SDP]) of the SIP payload message. The MIDCOM agent will parse the SDP section and use the MIDCOM protocol to (a) open pinholes (i.e., permit RTP/RTCP session tuples) in a middlebox implementing firewall service, or (b) create PORT bindings and appropriately modify the SDP content to permit the RTP/RTCP streams through a middlebox implementing NAT service. The MIDCOM protocol should be sufficiently rich and expressive to support the operations described under the timelines. The examples do not show the timers maintained by the agent to keep the middlebox policy rule(s) from timing out.

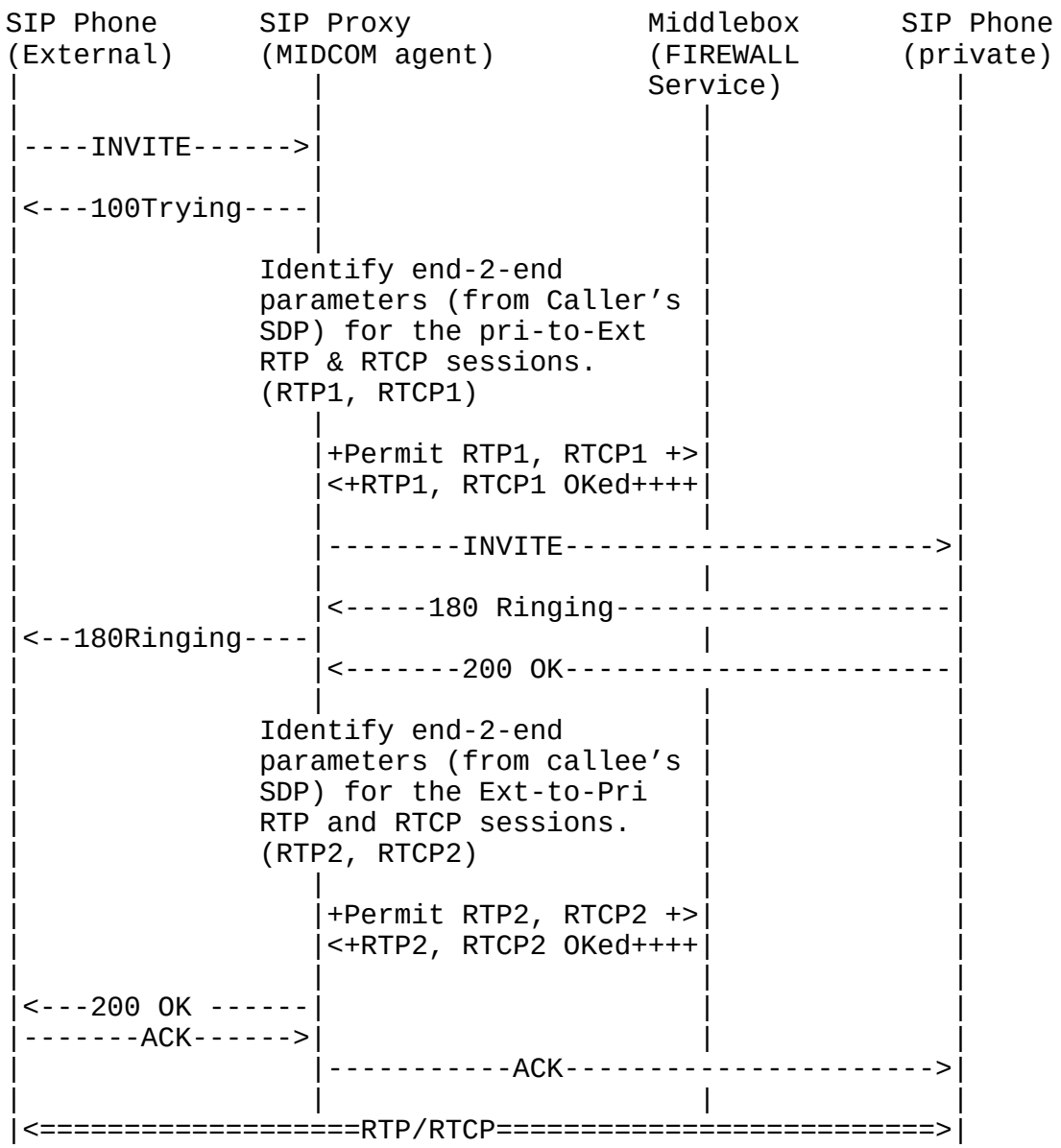
MIDCOM agent Registration and connectivity between the MIDCOM agent and the middlebox are not shown in the interest of restricting the focus of the MIDCOM transactions to enabling the middlebox to let the media stream through. MIDCOM PDP is also not shown in the diagram below or on the timelines for the same reason.

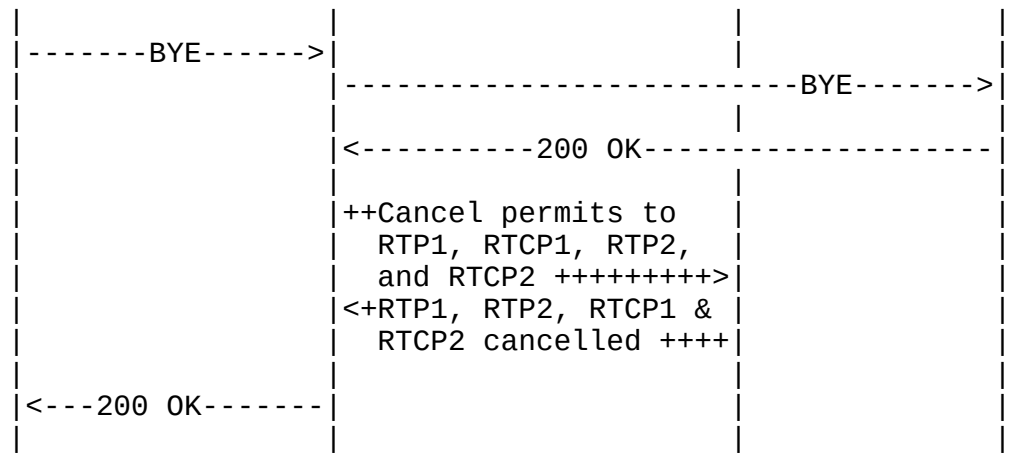
The following subsections illustrate a typical timeline sequence of operations that transpire with the various elements involved in a SIP telephony application path. Each subsection is devoted to a specific instantiation of a middlebox service - NAPT (refer [NAT-TERM], [NAT-TRAD]), firewall and a combination of both NAPT and firewall are considered.

### 7.1. Timeline flow - Middlebox implementing firewall service

In the following example, we will assume a middlebox implementing a firewall service. We further assume that the middlebox is pre-configured to permit SIP calls (destination TCP or UDP port number set to 5060) into the private phone. The following timeline illustrates the operations performed by the MIDCOM agent, to permit RTP/RTCP media stream through the middlebox.

The INVITE from the caller (external) is assumed to include the SDP payload. You will note that the MIDCOM agent requests the middlebox to permit the Private-to-external RTP/RTCP flows before the INVITE is relayed to the callee. This is because, in SIP, the calling party must be ready to receive the media when it sends the INVITE with a session description. If the called party (private phone) assumes this and sends "early media" before sending the 200 OK response, the firewall will have blocked these packets without this initial MIDCOM signaling from the agent.



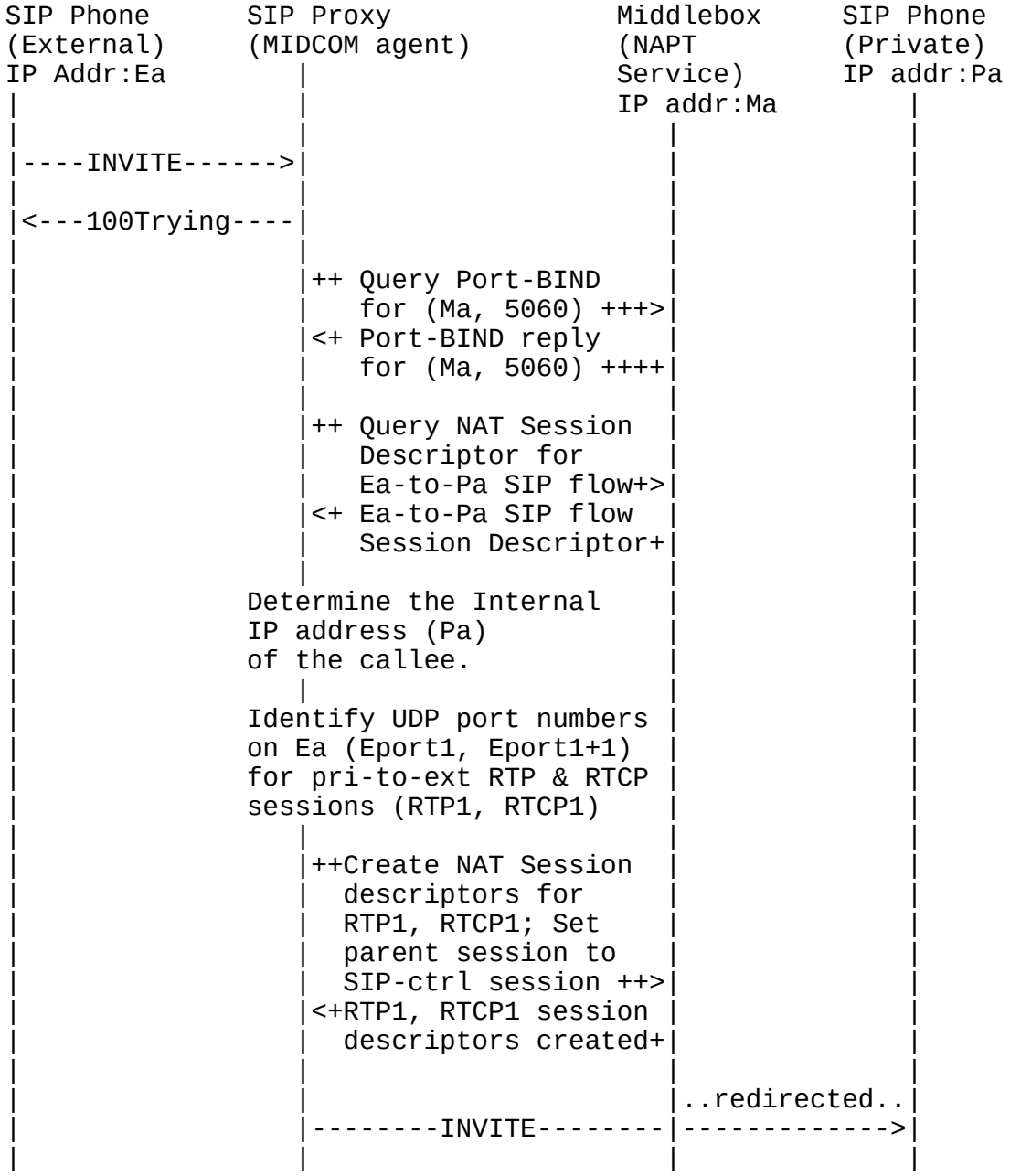


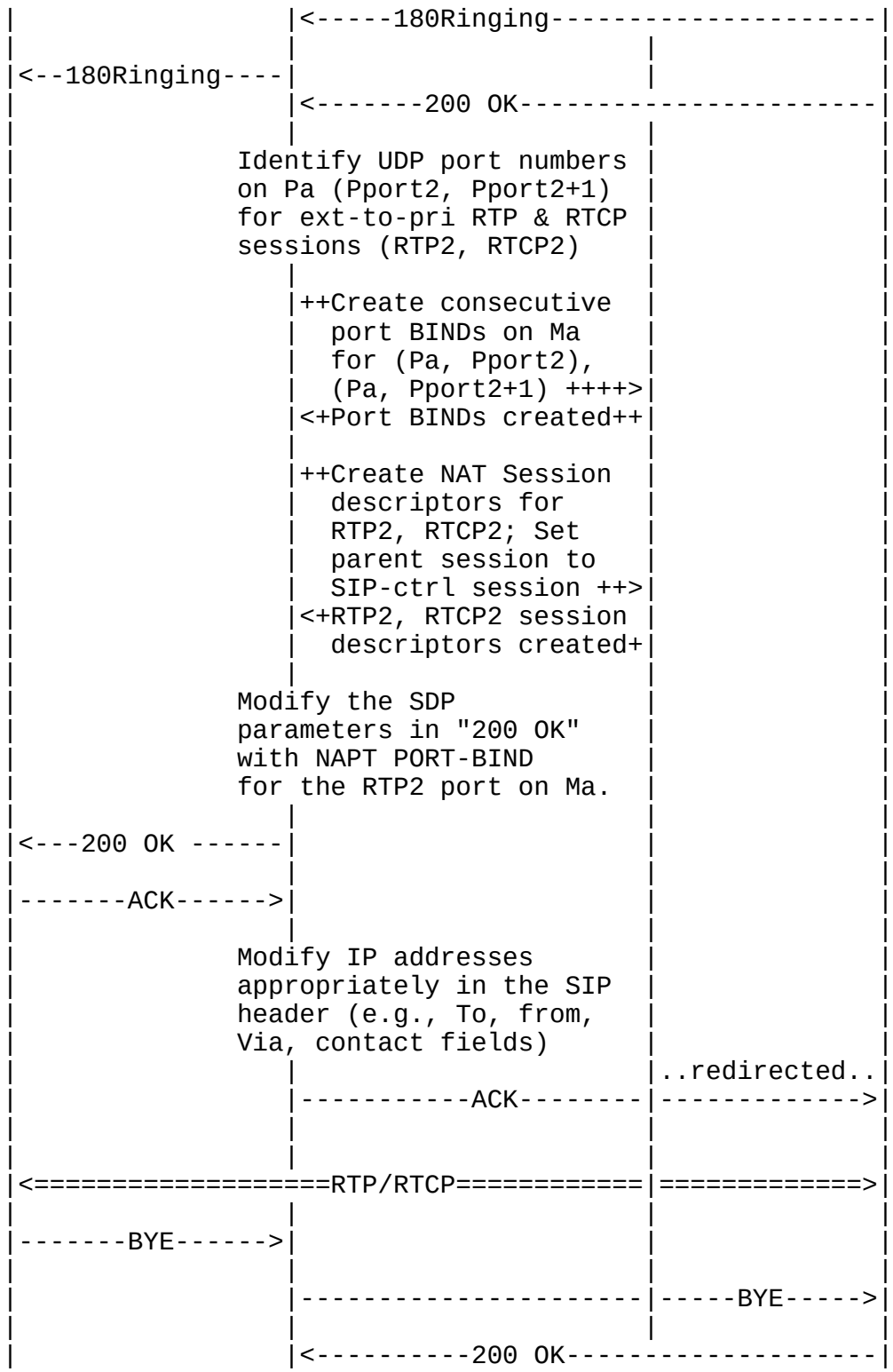

### 7.2. Timeline flow - Middlebox implementing NAPT service

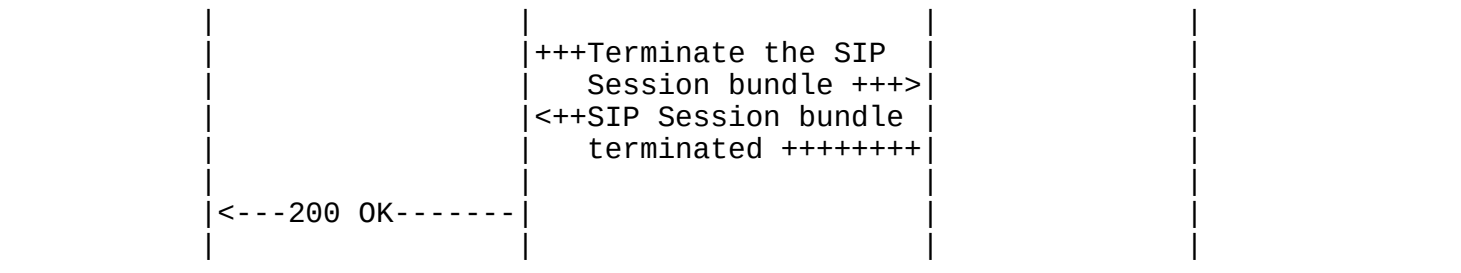
In the following example, we will assume a middlebox implementing NAPT service. We make the assumption that the middlebox is pre-configured to redirect SIP calls to the specific private SIP phone application. I.e., the INVITE from the external caller is not made to the private IP address, but to the NAPT external address. Let us say, the external phone's IP address is Ea, NAPT middlebox external Address is Ma, and the internal SIP phone's private address is Pa. SIP calls to the private SIP phone will arrive as TCP/UDP sessions, with the destination address and port set to Ma and 5060 respectively. The middlebox will redirect these datagrams to the internal SIP phone. The following timeline will illustrate the operations necessary to be performed by the MIDCOM agent to permit the RTP/RTCP media stream through the middlebox.

As with the previous example (section 7.1), the INVITE from the caller (external) is assumed to include the SDP payload. You will note that the MIDCOM agent requests the middlebox to create NAT session descriptors for the private-to-external RTP/RTCP flows before the INVITE is relayed to the private SIP phone (for the same reasons as described in section 7.1). If the called party (private phone) sends "early media" before sending the 200 OK response, the NAPT middlebox will have blocked these packets without the initial MIDCOM signaling from the agent. Also, note that after the 200 OK is received by the proxy from the private phone, the agent requests the middlebox to allocate NAT session descriptors for the external-to-private RTP2 and RTCP2 flows, such that the ports assigned on the Ma for RTP2 and RTCP2 are contiguous. The RTCP stream does not happen

with a non-contiguous port. Lastly, you will note that even though each media stream (RTP1, RTCP1, RTP2 and RTCP2) is independent, they are all tied to the single SIP control session, while their NAT session descriptors were being created. Finally, when the agent issues a terminate session bundle command for the SIP session, the middlebox is assumed to delete all associated media stream sessions automagically.




<-----180Ringing----------
<--180Ringing----
<-------200 OK----------
Identify UDP port numbers
on Pa (Pport2, Pport2+1)
for ext-to-pri RTP & RTCP
sessions (RTP2, RTCP2)
++Create consecutive
port BINDs on Ma
for (Pa, Pport2),
(Pa, Pport2+1) ++++>
<+Port BINDs created++
++Create NAT Session
descriptors for
RTP2, RTCP2; Set
parent session to
SIP-ctrl session ++>
<+RTP2, RTCP2 session
descriptors created+
Modify the SDP
parameters in "200 OK"
with NAPT PORT-BIND
for the RTP2 port on Ma.
<---200 OK ------
-------ACK------>
Modify IP addresses
appropriately in the SIP
header (e.g., To, from,
Via, contact fields)
..redirected..
-----------ACK--------
-------------->
<==========RTP/RTCP==========
==========>
-------BYE------>
----------
-----BYE----->
<----------200 OK----------

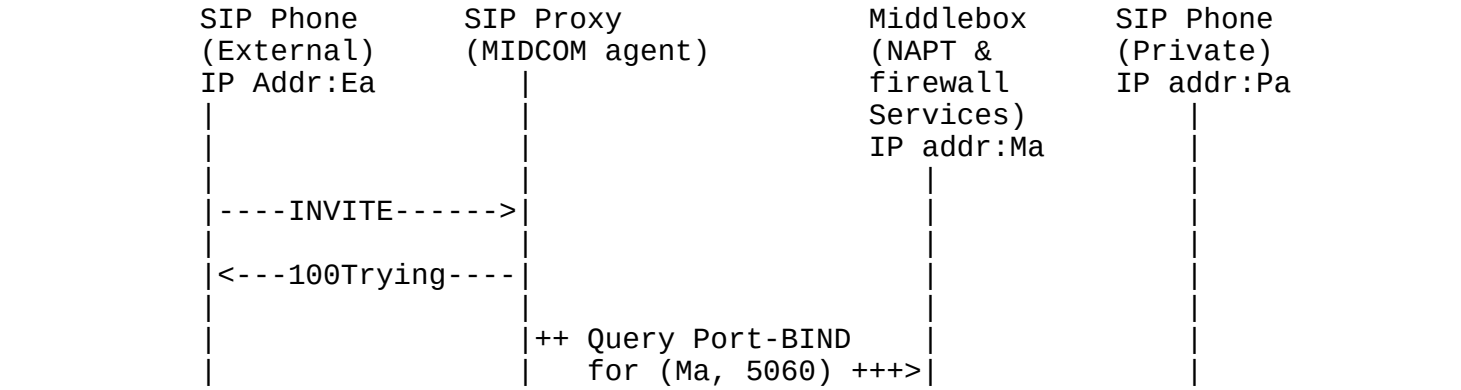


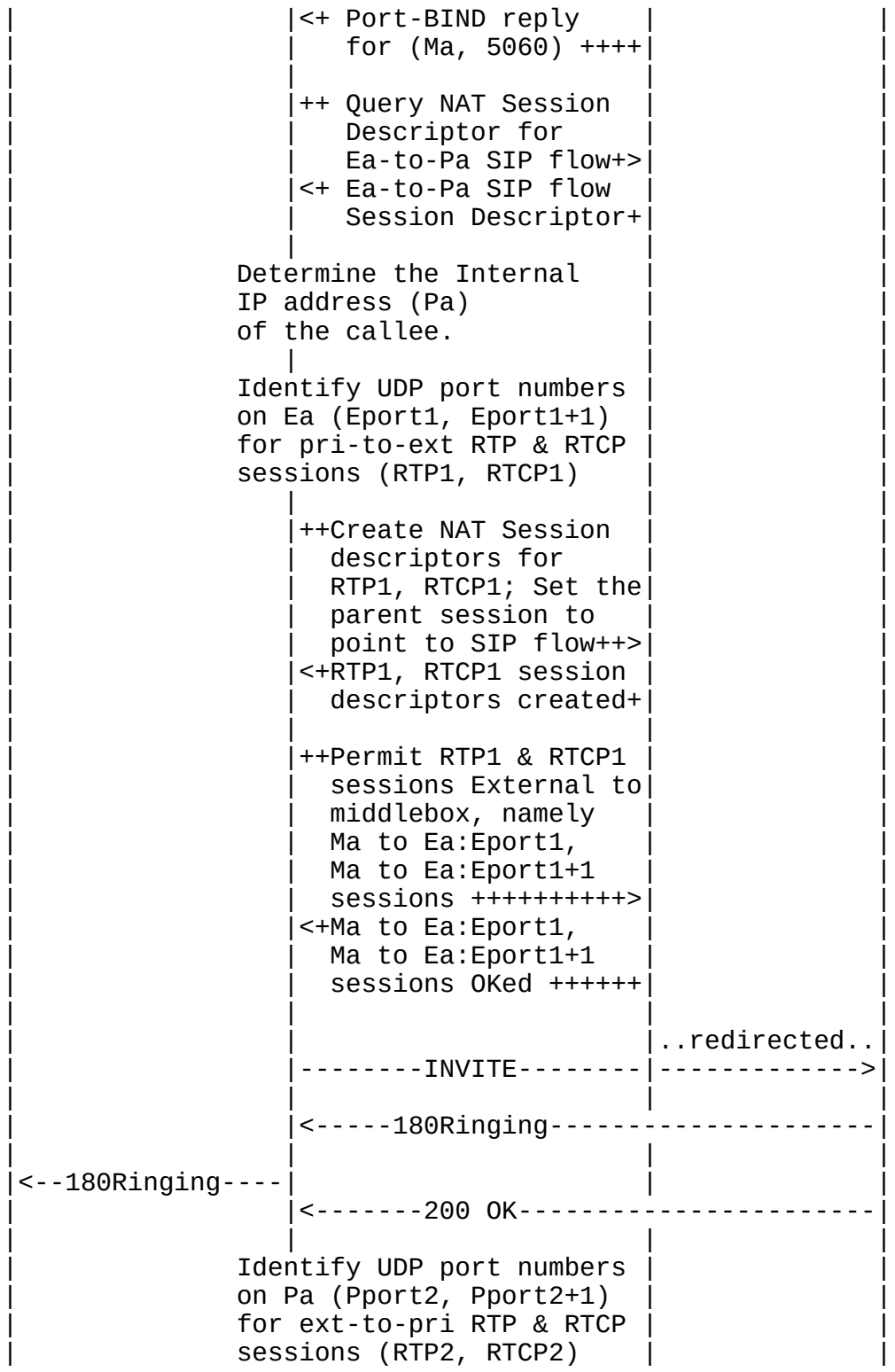

Legend: ++++ MIDCOM control traffic
---- SIP control traffic
==== RTP/RTCP media traffic

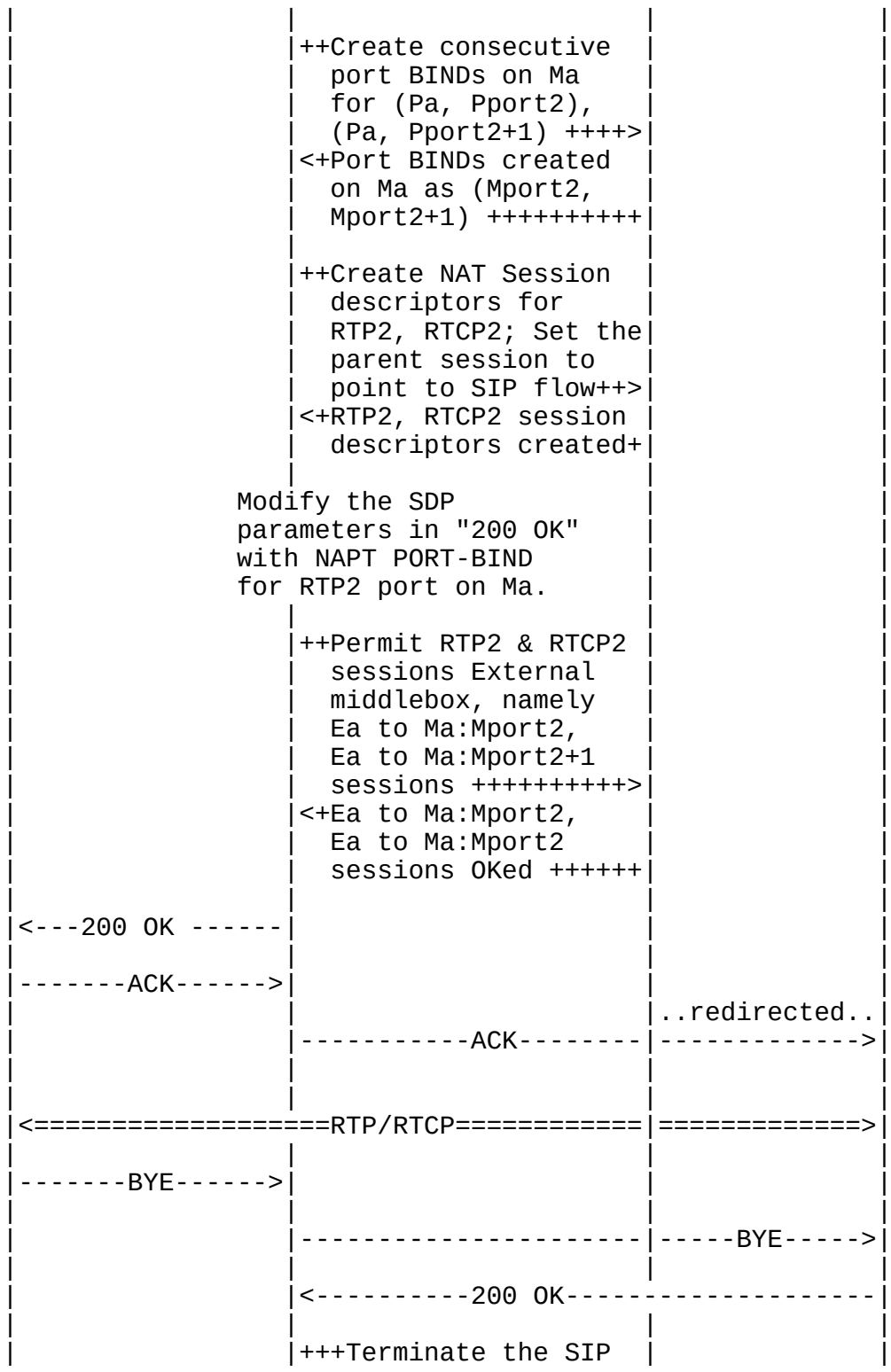
### 7.3. Timeline flow - Middlebox implementing NAPT and firewall

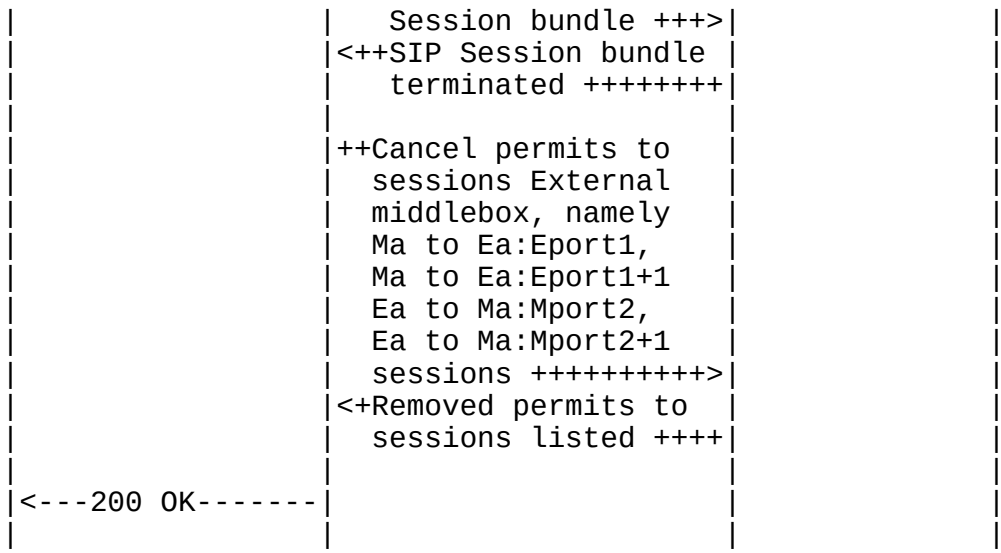
In the following example, we will assume a middlebox implementing a combination of a firewall and a stateful NAPT service. We make the assumption that the NAPT function is configured to translate the IP and TCP headers of the initial SIP session into the private SIP phone, and the firewall function is configured to permit the initial SIP session.

In the following time line, it may be noted that the firewall description is based on packet fields on the wire (ex: as seen on the external interface of the middlebox). In order to ensure correct behavior of the individual services, you will notice that NAT specific MIDCOM operations precede firewall specific operations on the MIDCOM agent. This is noticeable in the time line below when the MIDCOM agent processes the "200 OK" from the private SIP phone. The MIDCOM agent initially requests the NAT service on the middlebox to set up port-BIND and session-descriptors for the media stream in both directions. Subsequent to that, the MIDCOM agent determines the session parameters (i.e., the dynamic UDP ports) for the media stream, as viewed by the external interface and requests the firewall service on the middlebox to permit those sessions through.




|<+ Port-BIND reply
for (Ma, 5060) ++++|
|++ Query NAT Session
Descriptor for
Ea-to-Pa SIP flow+>|
|<+ Ea-to-Pa SIP flow
Session Descriptor+|
Determine the Internal
IP address (Pa)
of the callee.
Identify UDP port numbers
on Ea (Eport1, Eport1+1)
for pri-to-ext RTP & RTCP
sessions (RTP1, RTCP1)
|++Create NAT Session
descriptors for
RTP1, RTCP1; Set the|
parent session to
point to SIP flow++>|
|<+RTP1, RTCP1 session
descriptors created+|
|++Permit RTP1 & RTCP1
sessions External to|
middlebox, namely
Ma to Ea:Eport1,
Ma to Ea:Eport1+1
sessions ++++++++++>|
|<+Ma to Ea:Eport1,
Ma to Ea:Eport1+1
sessions OKed ++++++|
|..redirected..|
|--------INVITE--------|------------->|
|<-----180Ringing----------------------|
|<--180Ringing----|
|<-------200 OK-----------------------|
Identify UDP port numbers
on Pa (Pport2, Pport2+1)
for ext-to-pri RTP & RTCP
sessions (RTP2, RTCP2)


++Create consecutive port BINDs on Ma for (Pa, Pport2), (Pa, Pport2+1) ++++>
<+Port BINDs created on Ma as (Mport2, Mport2+1) ++++++++++
++Create NAT Session descriptors for RTP2, RTCP2; Set the parent session to point to SIP flow++>
<+RTP2, RTCP2 session descriptors created+
Modify the SDP parameters in "200 OK" with NAPT PORT-BIND for RTP2 port on Ma.
++Permit RTP2 & RTCP2 sessions External middlebox, namely Ea to Ma:Mport2, Ea to Ma:Mport2+1 sessions ++++++++++>
<+Ea to Ma:Mport2, Ea to Ma:Mport2 sessions OKed ++++++
<---200 OK ------
-------ACK------>
..redirected..
-----------ACK--------
<==========RTP/RTCP==========
==========>
-------BYE------>
-----BYE----->
<----------200 OK----------
+++Terminate the SIP




## 8.0. Operational considerations

### 8.1. Multiple MIDCOM sessions between agents and middlebox

A middlebox cannot be assumed to be a simple device implementing just one middlebox function and no more than a couple of interfaces. Middleboxes often combine multiple intermediate functions into the same device and have the ability to provision individual interfaces of the same device with different sets of functions and varied provisioning for the same function across the interfaces.

As such, a MIDCOM agent ought to be able to have a single MIDCOM session with a middlebox and use the MIDCOM interface on the middlebox to interface with different services on the same middlebox.

### 8.2. Asynchronous notification to MIDCOM agents

Asynchronous notification by the middlebox to a MIDCOM agent can be useful for events such as Session creation, Session termination, MIDCOM protocol failure, middlebox function failure or any other significant event. Independently, ICMP error codes can also be useful to notify transport layer failures to the agents.

In addition, periodic notification of various forms of data, such as statistics update, would also be a useful function that would be beneficial to certain types of agents.

### 8.3. Timers on middlebox considered useful

When supporting the MIDCOM protocol, the middlebox is required to allocate dynamic resources, as specified in policy rule(s), upon request from agents. Explicit release of dynamically allocated resources happens when the application session is ended or when a MIDCOM agent requests the middlebox to release the resource.

However, the middlebox should be able to recover the dynamically allocated resources, even as the agent that was responsible for the allocation is not alive. Associating a lifetime for these dynamic resources and using a timer to track the lifetime can be a good way to accomplish this.

### 8.4. Middleboxes supporting multiple services

A middlebox could be implementing a variety of services (e.g. NAT and firewall) in the same box. Some of these services might have inter-dependency on shared resources and sequence of operation. Others may be independent of each other. Generally speaking, the sequence in which these function operations may be performed on datagrams is not within the scope of this document.

In the case of a middlebox implementing NAT and firewall services, it is safe to state that the NAT operation on an interface will precede a firewall on the egress and will follow a firewall on the ingress. Further, firewall access control lists, used by a firewall, are assumed to be based on session parameters, as seen on the interface supporting firewall service.

### 8.5. Signaling and Data traffic

The class of applications the MIDCOM architecture addresses focus around applications that have a combination of, one or more, signaling and data traffic sessions. The signaling may be done out-of-band, using a dedicated stand-alone session or may be done in-band, within a data session. Alternately, signaling may also be done as a combination of both stand-alone and in-band sessions.

SIP is an example of an application based on distinct signaling and data sessions. A SIP signaling session is used for call setup between a caller and a callee. A MIDCOM agent may be required to examine/modify SIP payload content to administer the middlebox so as to let the media streams (RTP/RTCP based) through. A MIDCOM agent is not required to intervene in the data traffic.

Signaling and context specific Header information is sent in-band, within the same data stream for applications such as HTTP embedded applications, sun-RPC (embedding a variety of NFS apps), Oracle transactions (embedding oracle SQL+, MS ODBC, Peoplesoft) etc.

H.323 is an example of an application that sends signaling in both dedicated stand-alone sessions, as well as in conjunction with data. H.225.0 call signaling traffic traverses middleboxes by virtue of static policy, no MIDCOM control needed. H.225.0 call signaling also negotiates ports for an H.245 TCP stream. A MIDCOM agent is required to examine/modify the contents of the H.245 so that H.245 can traverse it.

H.245 traverses the middlebox and also carries Open Logical Channel information for media data. So, the MIDCOM agent is once again required to examine/modify the payload content needs to let the media traffic flow.

The MIDCOM architecture takes into consideration, supporting applications with independent signaling and data sessions as well as applications that have signaling and data communicated over the same session.

In the cases where signaling is done on a single stand-alone session, it is desirable to have a MIDCOM agent interpret the signaling stream and program the middlebox (that transits the data stream) so as to let the data traffic through uninterrupted.

## 9. Applicability Statement

Middleboxes may be stationed in a number of topologies. However, the signaling framework outlined in this document may be limited to only those middleboxes that are located in a DMZ (De-Militarized Zone) at the edge of a private domain, connecting to the Internet. Specifically, the assumption is that you have a single middlebox (running NAT or firewall) along the application route. Discovery of a middlebox along an application route is outside the scope of this document. It is conceivable to have middleboxes located between departments within the same domain or inside the service provider’s domain and so forth. However, care must be taken to review each individual scenario and determine the applicability on a case-by-case basis.

The applicability may also be illustrated as follows. Real-time and streaming applications, such as Voice-Over-IP, and peer-to-peer applications, such as Napster and Netmeeting, require administering firewalls and NAT middleboxes to let their media streams reach hosts inside a private domain. The requirements are in the form of

establishing a "pin-hole" to permit a TCP/UDP session (the port parameters of which are dynamically determined) through a firewall or retain an address/port bind in the NAT device to permit sessions to a port. These requirements are met by current generation middleboxes using adhoc methods, such as embedding application intelligence within a middlebox to identify the dynamic session parameters and administering the middlebox internally as appropriate. The objective of the MIDCOM architecture is to create a unified, standard way to exercise this functionality, currently existing in an ad-hoc fashion, in some of the middleboxes.

By adopting MIDCOM architecture, middleboxes will be able to support newer applications they have not been able to support thus far. MIDCOM architecture does not, and must not in anyway, change the fundamental characteristic of the services supported on the middlebox.

Typically, organizations shield a majority of their corporate resources (such as end-hosts) from visibility to the external network by the use of a De-Militarized Zone (DMZ) at the domain edge. Only a portion of these hosts are allowed to be accessed by the external world. The remaining hosts and their names are unique to the private domain. Hosts visible to the external world and the authoritative name server that maps their names to network addresses are often configured within a DMZ (De-Militarized Zone) in front of a firewall. Hosts and middleboxes within DMZ are referred to as DMZ nodes.

Figure 4 below illustrates the configuration of a private domain with a DMZ at its edge. Actual configurations may vary. Internal hosts are accessed only by users inside the domain. Middleboxes, located in the DMZ may be accessed by agents inside or outside the domain.

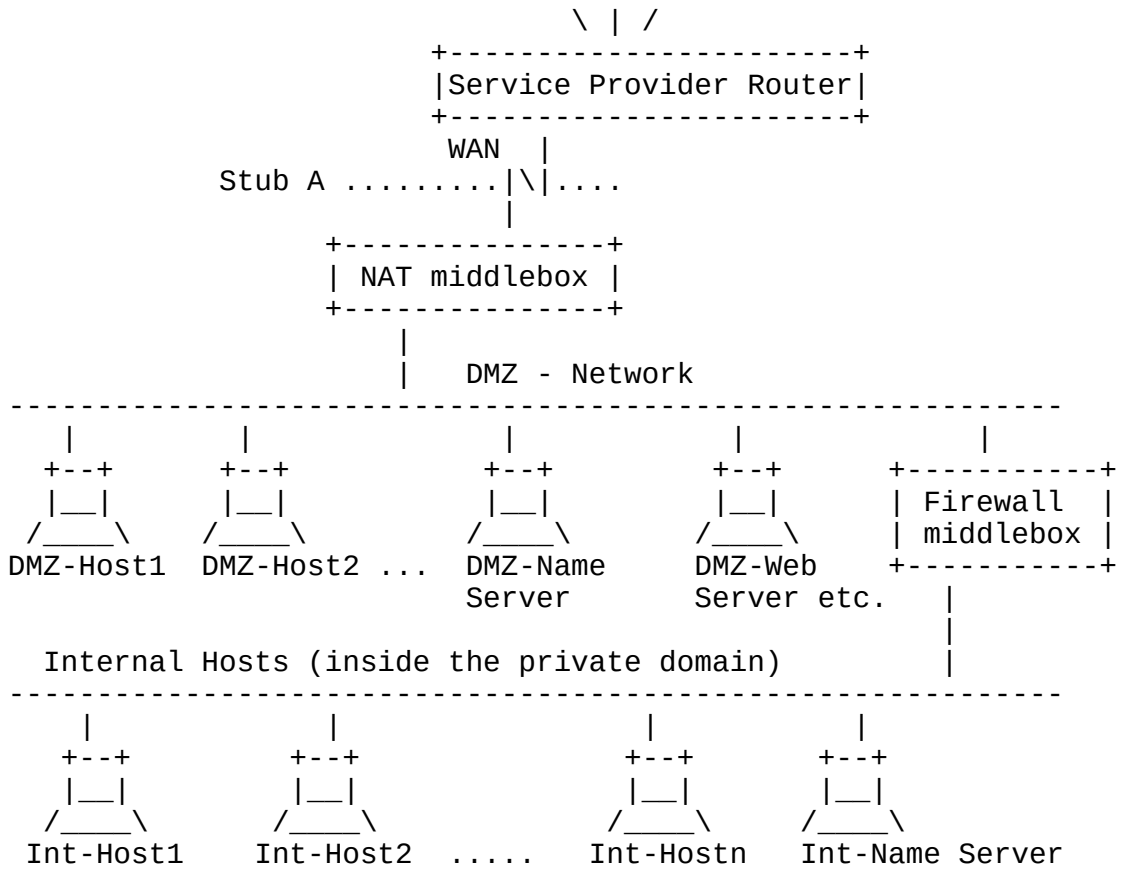



Figure 4: DMZ network configuration of a private domain.

## 10. Acknowledgements

The authors wish to thank Christian Huitema, Joon Maeng, Jon Peterson, Mike Fisk, Matt Holdrege, Melinda Shore, Paul Sijben, Philip Mart, Scott Brim and Richard Swale for their valuable critique, advice and input on an earlier rough version of this document. The authors owe special thanks to Eliot Lear for kick-starting the e-mail discussion on use-case scenarios with a SIP application flow diagram through a middlebox. Much thanks to Bob Penfield, Cedric Aoun, Christopher Martin, Eric Fleischman, George Michaelson, Wanqun Bao, and others in the MIDCOM work group for their very detailed feedback on a variety of topics and adding clarity to the discussion. Last, but not the least, the authors owe much thanks to Mark Duffy, Scott Brim, Melinda Shore and others for their help with terminology definition and discussing the embedded requirements within the framework document.

## 11. Security Considerations

Discussed below are security considerations in accessing a middlebox. Without MIDCOM protocol support, the premise of a middlebox operation fundamentally requires the data to be in the clear, as the middlebox needs the ability to inspect and/or modify packet headers and payload. This compromises the confidentiality requirement in some environments. Further, updating transport headers and rewriting application payload data, in some cases, by NAT prevents the use of integrity protection on some data streams traversing NAT middleboxes. Clearly, this can pose a significant security threat to the application in an untrusted transport domain.

The MIDCOM protocol framework removes the need for a middlebox to inspect or manipulate transport payload. This allows applications to better protect themselves end-to-end with the aid of a trusted MIDCOM agent. This is especially the case when the agent is a resident on the end-host. When an agent has the same end-to-end ability as the end-host to interpret encrypted and integrity protected data, transiting a middlebox can be encrypted and integrity protected. The MIDCOM agent will still be able to interpret the data and simply notify the middlebox of open holes, install NAT table entries, etc. Note, however, the MIDCOM framework does not help with the problem of NAT breaking IPsec since in this case the middlebox still modifies IP and transport headers.

Security between a MIDCOM agent and a middlebox has a number of components. Authorization, authentication, integrity and confidentiality. Authorization refers to whether a particular agent is authorized to signal a middlebox with requests for one or more applications, adhering to a certain policy profile. Failing the authorization process might indicate a resource theft attempt or failure due to administrative and/or credential deficiencies. In either case, the middlebox should take the proper measures to audit/log such attempts and consult its designated MIDCOM PDP for the required action if the middlebox is configured with one. Alternatively, the middlebox may resort to a default service deny policy when a MIDCOM agent fails to prompt the required credentials. Section 6 discusses the middlebox to MIDCOM PDP interactions in view of policy decisions.

Authentication refers to confirming the identity of an originator for all datagrams received from the originator. Lack of strong credentials for authentication of MIDCOM messages between an agent and a middlebox can seriously jeopardize the fundamental service rendered by the middlebox. A consequence of not authenticating an agent would be that an attacker could spoof the identity of a "legitimate" agent and open holes in the firewall. Another would be

that it could otherwise manipulate the state on a middlebox, creating a denial-of-service attack by closing needed pinholes or filling up a NAT table. A consequence of not authenticating the middlebox to an agent is that an attacker could pose as a middlebox and respond to NAT requests in a manner that would divert data to the attacker. Failing to submit the required/valid credentials, once challenged, may indicate a replay attack, in which case a proper action is required by the middlebox such as auditing, logging, or consulting its designated MIDCOM PDP to reflect such failure. A consequence of not protecting the middlebox against replay attacks would be that a specific pinhole may be reopened or closed by an attacker at will, thereby bombarding end hosts with unwarranted data or causing denial of service.

Integrity is required to ensure that a MIDCOM message has not been accidentally or maliciously altered or destroyed. The result of a lack of data integrity enforcement in an untrusted environment could be that an imposter will alter the messages sent by an agent and bring the middlebox to a halt or cause a denial of service for the application the agent is attempting to enable.

Confidentiality of MIDCOM messages ensure that the signaling data is accessible only to the authorized entities. When a middlebox agent is deployed in an untrusted environment, lack of confidentiality will allow an intruder to perform traffic flow analysis and snoop the middlebox. The intruder could cannibalize a lesser secure MIDCOM session and destroy or compromise the middlebox resources he uncovered on other sessions. Needless to say, the least secure MIDCOM session will become the achilles heel and make the middlebox vulnerable to security attacks.

Lastly, there can be security vulnerability to the applications traversing a middlebox when a resource on a middlebox is controlled by multiple external agents. A middlebox service may be disrupted due to conflicting directives from multiple agents associated with different middlebox functions but applied to the same application session. Care must be taken in the protocol design to ensure that agents for one function do not abruptly step over resources impacting a different function. Alternately, the severity of such manifestations could be lessened when a single MIDCOM agent is responsible for supporting all the middlebox services for an application, due to the reduced complexity and synchronization effort in managing the middlebox resources.

References

[SIP] Rosenberg, J., Shulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E., "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[SDP] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[H.323] ITU-T Recommendation H.323. "Packet-based Multimedia Communications Systems," 1998.

[RTP] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[RTSP] Schulzrinne, H., Rao, A. and R. Lanphier: "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[FTP] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, October 1985.

[NAT-TERM] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[NAT-TRAD] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[NAT-PT] Tsirtsis, G. and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[IPsec-AH] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[IPsec-ESP] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[TLS] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[POL-TERM] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J. and S. Waldbusser, "Terminology for Policy-Based Management", RFC 3198, November 2001.

[REQMTS] Swale, R. P., Mart, P. A., Sijben, P., Brim, S. and M. Shore, "Middlebox Communications (midcom) Protocol Requirements", RFC 3304, August 2002.

## Authors' Addresses

Pyda Srisuresh
Kuokoa Networks, Inc.
475 Potrero Ave.
Sunnyvale, CA 94085
EMail: srisuresh@yahoo.com

Jiri Kuthan
Fraunhofer Institute FOKUS
Kaiserin-Augusta-Allee 31
D-10589 Berlin, Germany
EMail: kuthan@fokus.fhg.de

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
U.S.A.
EMail: jdrosen@dynamicsoft.com

Andrew Molitor
Aravox technologies
4201 Lexington Avenue North, Suite 1105
Arden Hills, MN 55126
U.S.A.
voice: (651) 256-2700
EMail: amolitor@visi.com

Abdallah Rayhan
WINCORE Lab
Electrical and Computer Engineering
Ryerson University
350 Victoria Street
Toronto, ON M5B 2K3
EMail: rayhan@ee.ryerson.ca, ar_rayhan@yahoo.ca

## Full Copyright Statement



## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.