A NAT and Firewall signaling framework for the Internet

Cédric Aoun

Thèse

Présentée pour l'obtention du titre de DOCTEUR de
L'École Nationale Supérieure des Télécommunications

Discipline : **Informatique et Réseaux**

Cédric AOUN

Plan de signalisation Internet pour l'interfonctionnement entre NAT et Firewall

Soutenue le 12 décembre 2005 devant le jury composé de :

Elie Najm Professeur à l'ENST Paris	**Président**
Nazim Agoulmine Professeur à l'Université d'EVRY	**Rapporteurs**
Omar Cherkaoui Professeur à l'Université du Québec à Montréal	
Olivier Paul Maître de conférences à INT-EVRY	**Examinateurs**
Raouf Boutaba Professeur à l'Université de Waterloo	
Ahmed Serhrouchni	**Directeur de Thèse**

Résumé des travaux de recherche et de ce manuscrit

Préface

Ce document résume le fruit des recherches et contributions effectuées par le doctorant dans le but de résoudre les problèmes provoqués par l' emploi des NATs et Firewalls en interaction avec plusieurs applications utilisées par un nombre croissant d' internautes dans le monde.

Problématiques

De plus en plus d' opérateurs et d' entreprises migrent leurs applications multimédia (voix et vidéo) vers les réseaux de données, principalement les réseaux IP ; certains réseaux ont déployé des services voix et vidéo natifs dans des réseaux ATM mais en moindre nombre que les réseaux multimédia sur IP.
Les déploiements d' applications multimédia sur IP ont principalement commencé au sein même des entreprises. Ainsi un employé d' une entreprise appelant son collègue génère un flux VoIP qui aboutit sur le terminal VoIP de ce dernier. Auparavant, pour les appels inter-entreprises, une interconnexion de type RNIS était utilisée (et donc sans aucune communication VoIP entre les entreprises).

Avec le développement exponentiel du haut débit (principalement l' ADSL), un nombre croissant de fournisseurs de services déploie des solutions VoIP pour leurs abonnés professionnels ou résidentiels. Certains opérateurs commencent à offrir des services VoIP à des abonnés interconnectés en direct (abonnés ADSL) ou bien à des abonnés n'ayant aucun accès direct au fournisseur de services.

Cependant les interférences avec les mécanismes de sécurité et de traduction d' adresse IP déployés dans les réseaux empêchent toute utilisation de ces nouveaux services.
Ainsi M. Dupont appelant M. Doe dans un autre réseau IP, génère un flux VoIP traversant les pare-feux (firewall, [1]) et les traducteurs d' adresse (NAT) ([2], [3]) de l' entreprise de M. Dupont.

Dans la suite de cette section, nous aborderons plusieurs exemples concrets d' applications ayant des interactions problématiques avec les NATs et les firewalls.

Migration vers la Voix sur IP des PME et des grandes entreprises

Le marché des PBX entreprises est en plein renouvellement; toutes les entreprises recherchent des IP PBX avec de nouveaux terminaux intégrant des services Voix et des services d' échanges d' information (annuaire, client http, web cam, etc). Ces nouveaux déploiements sont aussi impactés par les NAT et les firewalls, dans les scénarios mentionnés ci-dessus.

Déploiement du service Centrex dans le monde

En Amérique du Nord, il est très fréquent que les abonnés professionnels externalisent leur service téléphonique pour des raisons de coûts d' équipements et de maintenance. Le service fourni à ces abonnés est appelé un service centrex ; ainsi en Amérique du Nord, plus de 30 millions d' abonnés bénéficient de ce service.

La vague centrex a aussi envahi le reste du monde ; on compte plus de 4 millions d' abonnés centrex au UK (BT feature net), une centaine de milliers d' abonnés en France (France Telecom et nouveaux opérateurs) ; dans le Sud-Est asiatique et l' Océanie plusieurs millions d' abonnés sont déjà recensés. La plupart des fournisseurs du service centrex migrent vers la VoIP ; cette migration devra faire face aux firewalls et aux traducteurs d' adresse IP (NAT).

Activités issues de la délocalisation

Le haut débit permettra la délocalisation d' activités sociales, et réciproquement la délocalisation croissante accélérera le déploiement du haut débit. Les activités les plus connues issues de la délocalisation sont à l' heure actuelle le télétravail, l' enseignement à distance, la télésurveillance (VoIP et vidéo sur IP) et la médecine à distance. Toutes ces activités sont impactées par l' architecture actuelle des réseaux déployant des NAT et des firewalls.

En conclusion, nous pouvons nous interroger sur l' avenir des services multimédias sur IP : le multimédia est-il destiné à ne pas être déployé dans l' architecture des réseaux actuels ?
Le problème principal du multimédia est en effet dû au fait que chaque application a une session de signalisation et une session d' échange d' informations (voix, vidéo, graphisme, etc). Ainsi la session de signalisation informe l' application distante de l' adresse et du port de transport utilisés pour recevoir l' information.

Au cas où un NAT serait déployé à la périphérie du réseau, l' hôte de l' application ne serait pas au courant du NAT et ne connaîtrait pas son adresse traduite : en conséquence, l' application ne fonctionnerait pas (l' hôte de l' application ne fournirait qu'une adresse ayant une signification locale et ne pouvant être routée).

Dans le cas où un firewall serait déployé à la périphérie du réseau, ce firewall bloquerait tous les flux prohibés par l' administrateur réseau. En effet,un firewall est un équipement qui

identifie des flux au moyen de filtres (5 tuple ou autre lorsque l' information utile n'est pas transportée par un protocole de transport (IPSEC AH\ESP, ICMP, IGMP, RSVP, etc)).
Un traitement est appliqué aux flux identifiés : le firewall accepte de router les paquets du flux ou de les bloquer (avec ou sans logs). Dans le cas d' applications multimédia, le firewall n'a aucune information sur les flux multimédia (les flux de données multimédia ne sont pas identifiables à l' avance avec une granularité très fine, plusieurs ports de transport doivent être permis) ; par conséquent soit l' application ne fonctionnera pas, soit elle fonctionnera mais aux dépens de la sécurité du réseau.

Des solutions doivent ainsi être définies pour résoudre ces problèmes, ce que les travaux de recherche du doctorant ont permis.

Objectifs des travaux de recherche du doctorant

- Obj1) Analyse des interférences des NAT et Firewalls avec les applications de l' Internet
- Obj2) Etat de l' art des solutions aux problèmes des NAT et Firewalls en octobre 2002
- Obj3) Proposition d' une architecture réseau utilisant des protocoles de signalisation des Firewalls et NAT. Un des principes fondamentaux de cette architecture est son indépendance du type de topologie de réseau.
- Obj4) Ingéniérie de la nouvelle architecture proposée dans le cadre de nouveaux réseaux (ainsi que les réseaux existants)
- Obj5) Définition de modèles d' implémentation de la solution proposée :
  - Firewall implémentant le protocole de signalisation proposé pour les NATs et Firewall
  - Implémentation du protocole de signalisation proposé sur un hôte ayant un client VoIP SIP [4]
  - Implémentation du protocole de signalisation proposé sur un hôte ayant un client VoIP SIP ainsi qu'un pare-feux intégré

Résultats et contributions des travaux de recherche du doctorant

- Res1) et Res2) :
  - Le doctorant a effectué une analyse détaillée des problèmes des NAT et Firewalls, ainsi un modèle de catégorisations de problèmes a été défini. De plus le doctorant a rédigé une spécification au sein du groupe d' études V6OPS de l' IETF détaillant tous les problèmes des traducteurs d' adresse et protocoles (bien que le document ne concerne que le NAT-PT [5]).
  - Le doctorant a contribué aux organismes Packet-Cable [6] et ITU-T [7] durant leurs investigations sur les problèmes des NATs et Firewalls dans le déploiement de leur architecture de solutions multimédias.
  - Une publication [8], expliquant ces problèmes et ainsi que l' état de l' art des solutions existantes à la conférence Contel 2005
- Res3) :
  - Les résultats des objectifs de recherche 1 et 2 ont amené le doctorant à déduire que la solution aux problèmes des NATs et Firewalls est d' utiliser un protocole de signalisation de ces derniers.
  - Le doctorant a contribué à la spécification, au sein du groupe d' étude MIDCOM de l' IETF, d' une solution intermédiaire (STUN [9]) ne nécessitant pas la modification des NATs (dans un nombre assez conséquent d' implémentations)
  - Une analyse des mécanismes fondamentaux des protocoles de signalisation (ainsi que des protocoles existants [10–13]) a été réalisée par le doctorant, dans laquelle celui-ci a démontré qu'une technologie de signalisation "Path-Directed" (ainsi dénommée par le doctorant) était nécessaire. Cette architecture a été définie dans le cadre des activités du groupe d' étude NSIS WG de l' IETF. Cependant, celui-ci n'a pas mené de recherche au niveau de l' architecture globale du réseau. Le doctorant a été l un des principaux membres actifs de ce groupe d' étude durant la phase de conception de la pile protocolaire. Le doctorant est un co-auteur de deux documents [14, 15] du groupe d' étude.

- Le doctorant a défini toute l' architecture tout en analysant les concepts fondamentaux des mécanismes de signalisation nécessaires à la résolution des problèmes des NATs et Firewall. La conception de cette architecture a été faite en prenant en compte les applications utilisatrices de la solution, ainsi quelques divergences existent entre la solution définie à l' IETF [16] et celle proposée par le doctorant.
  L' IETF étant un organisme de standardisation de protocoles, celui-ci ne prend pas en compte certains facteurs liés à l' architecture globale des réseaux dans laquelle sont déployés les protocoles. Cette divergence est discutée dans le manuscrit de la thèse.
  * Le protocole de la couche commune de la nouvelle pile protocolaire, "Path-Directed Message Transfer Protocol" (PDMTP), est le pilier de l' architecture. Il est similaire au protocole GIST [17] du NSIS WG
  * Le protocole de signalisation de NAT et Firewall, "Path-Directed NAT and Firewall Signaling" (PDNFS), est similaire au protocole NATFW NSLP [14]

- Res4):
  - Le doctorant a analysé l' impact des protocoles proposés sur les règles d' ingénierie des réseaux dans lesquels sera utilisée l' architecture proposée
  - Analyse des performances des protocoles proposés et impact sur les délais d' établissement de session des applications temps-réels
  - Résolution des problèmes de routage asymétrique avec les pare-feux (publication à la conférence CFIP05 [18])
  - Elaboration du cahier de charge de l' infrastructure de sécurité réseau
  - Définition d' extensions de certificat pour sécuriser la découverte de Firewalls et NAT dans les réseaux (publication à la conférence ASWN05 [19])
  - Utilisation d' un protocole de stimulation, par des entités applicatives (proxy d' applications), des routeurs du réseau pour utiliser les protocoles définis par le doctorant (publication à la conférence IPOM04 [20]) ; cette solution s'avère utile lorsque les hôtes applicatifs ne peuvent pas évoluer pour intégrer les protocoles proposés

  - Déploiement de la solution dans les réseaux IPv6

- Res5) :
  - Proposition d' un modèle d' intégration de PDMTP et PDNFS avec le Firewall PF [21] d' OpenBSD
  - Définition des API et des interactions avec les blocs fonctionnels
  - Proposition d' un modèle d' intégration de PDMTP et PDNFS avec un client SIP
    * Définition des API et des interactions avec les blocs fonctionnels
  - Proposition d' un modèle d' intégration de PDMTP et PDNFS avec un client SIP résidant sur une machine intégrant un pare-feux
    * Définition des API et des interactions avec les blocs fonctionnels

Bref descriptif du manuscrit

Ce manuscrit a été rédigé avec LaTeX et compilé avec pdfLatext; ceci a permis la conception d' un document ayant des liens actifs (dans sa version électronique bien évidemment). De plus l' index indique le numéros de la page dans laquelle le thème cité est défini.

Le manuscrit contient huit chapitres:

- Le chapitre 1 résume l'objectif de la thèse ainsi que les travaux effectués
- Le chapitre 2 décrit les problèmes des NAT et Firewall avec les applications ainsi que les solutions existantes et potentielles. Ce chapitre offre une vision exhaustive au lecteur des problèmes existants et démontre, par une comparaison entre les solutions existantes et potentielles, que la solution à adopter consiste à utiliser un protocole de signalisation. Ce dernier est utilisé entre les entités ayant connaissance des besoins des applications et les pare-feux et traducteurs d' adresses.
- Dans le chapitre 3, sont étudiés les différents types de protocoles de signalisation pouvant être utilisés sur l' Internet. Ces types de protocoles sont ainsi scindés en deux classes:
  - les protocoles nomms "targeted signaling protocols",

– et les protocoles identifiés comme "Path-Directed Signaling Protocols" (PDSP).

Ces deux classes de protocoles sont étudiées en détail et le doctorant démontre que le protocole de signalisation essentielle pour résoudre les problèmes de NAT et Firewall doit avoir les propriétés des protocoles "Path-Directed Signaling Protocols".
L' analyse détaillée des PDSPs indique qu'un modèle de signalisation à deux strates est nécessaire à ces protocoles afin de permettre la réutilisation des fonctions communes à tous les protocoles issus de cette famille. Ainsi le doctorant analyse:

– la séparation des fonctionnalités,

– et leur regroupement en une strate commune appelée "Common Path-Directed Signaling Layer" (CPDSL),

– et la couche protocolaire "Path-Directed Signaling Application Protocol" (PDSAP) qui est spécifique à chacun des PDSPs.

- Dans le chapitre 4, le doctorant analyse en détail les fonctionnalités nécessaires pour le protocole de la couche commune et définit le "Path-Directed Message Transfer Protocol" (PDMTP). Ce travail a été effectué en parallèle avec les travaux du doctorant au sein du comité de standardisation NSIS de l' IETF, dans le cadre de la conception du protocole GIST. Le protocole PDMTP permet la découverte des nœuds voisins et le transfert de message de proche en proche jusqu'au destinataire du message.

- Dans le chapitre 5, le doctorant propose un protocole de signalisation de pare-feux et de traducteur d' adresse utilisant la couche protocolaire commune CPDSL et son protocole PDMTP. L' analyse effectuée compare l' utilisation d' un seul ou de deux protocoles distincts et conclut qu'un seul protocole est nécessaire, le protocole "Path-Directed NAT and Firewall Signaling" (PDNFS). Les besoins fonctionnels ainsi que le mode opératoire sont étudiés en détail et permettent au doctorant de proposer une sémantique pour le protocole. Celle-ci a été utilisée pour la définition du protocole "NAT and Firewall NSLP" au sein du comité de standardisation NSIS de l' IETF. Il est à noté que le doctorant a participé, en tant que co-auteur, à la rédaction de la spécification de ce protocole.

- Dans le chapitre 6, le doctorant analyse les problématiques d' ingénierie et de déploiement de sa nouvelle solution de signalisation. Ainsi l' impact temps réel de la solution est étudié, ainsi que la problématique du routage asymétrique, le déploiement dans les réseaux IPv6 et l' intégration de la solution dans les architectures de sécurité existantes.

- Dans le chapitre 7, le doctorant propose des modèles de mises en place de sa proposition dans des implémentations de pare-feux et traducteur d' adresse (PF de OpenBSD) ainsi que dans des hôtes ayant des applications impactées par ces derniers (le candidat discute de la VoIP pilotée par le protocole SIP).

- Dans le chapitre 8, le doctorant conclut le manuscrit en résumant ses travaux de recherche et les solutions apportées, ainsi qu'en décrivant d' autres activités de recherche qui seront nécessaires pour un déploiement plus étendu de ses propositions.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank:

- Professor Ahmed Serhrouchni for accepting to be my supervisor and for his guidance
- Professor Nazim Agoulmine from the EVRY University (France) and Professor Omar Cherkaoui from the Québec University (Canada), for accepting to be my thesis reviewers
- Olivier Paul from the INT (France), Professor Raouf Boutaba from the Waterloo University (Canada) and Professor Elie Najm from ENST Paris (France), for accepting to be on my thesis jury board
- My previous manager at Nortel and friend Patrick Bradd for assigning me the responsibility to solve applications' NAT and Firewall traversal issues in October 2000 . Without that assignment I would not have been curious to solve NAT and Firewall issues and propose a global solution for the Internet

Several people deserve recognition for: the hours of discussions and brainstorming that I had with them, as well as the effort spent and influence on the NSIS IETF specifications on which we have worked together; thank you Elwyn Davies, Xiaoming Fu, Robert Hancock, John Loughney, Andrew McDonald, Professor Henning Schulzrinne, Martin Stiemerling and Hannes Tschofenig.

Many thanks to Martin Stiemerling for his early comments on the manuscript and to Elwyn Davies for his very detailed review of the first complete version of the manuscript.

Special thanks to, my brother and grand-parents for their support during the last months of the manuscript writing; my wife Caroline and daughter Margaux for their support and understanding of the long nights and week-ends spent in the town-hall and school libraries to accomplish this work.

Finally I would like to dedicate my thesis to my parents who sacrificed several years of their lives in ensuring me a good education, I hope I have made them proud.

Abstract

In the early 1980s when IPv4 was being designed a 32 bit address space was believed to be sufficient for the Internet and the exponential growth of the Internet was definitely not foreseen. A decade later, the Internet research community started investigations for solutions to address IPv4's address space limitations. Two main solutions were proposed: development of a new Internet Protocol, IPv6, and Network Address Translators (NATs) for multiplexing private IPv4 addresses onto globally routable IPv4 addresses. During the same period network perimeter security started to be massively deployed in the Internet with its key element being Firewalls. Although believed to be useful for the Internet, both NATs and Firewalls had unforeseen impacts; several applications were impacted including Voice over IP, Video over IP, Instant Messaging and peer to peer applications which are critical for today's social life and economy. Several solutions have been proposed starting the middle of the 1990s with the deployment of SOCKS and application proxies, which allowed continued usage of network perimeter security while preserving application friendliness. In the late 1990s, Application Layer Gateways (ALGs) appeared in the Internet to solve the applications' traversal of NATs and Firewalls. All these solutions had various drawbacks and led the Internet research community and the industry to investigate alternative techniques such as reflector servers, media proxies and NAT and Firewall signaling protocols.

The work described in this thesis has been focused on analyzing NAT and Firewall impacts on the Internet's applications, analysis of existing solutions and a proposal for a novel NAT and Firewall signaling framework using a Path-Directed NAT and Firewall signaling protocol that would allow secured and reliable traversal of NATs and Firewalls by application protocols with minimal application awareness of network topology.

The overall work accomplished within the thesis period covers: analysis of existing NAT and Firewall traversal solutions, analysis of a NAT and Firewall signaling architecture framework, a NAT and Firewall signaling protocol proposal with deployment considerations (migration and required network infrastructure requirements to support the proposal) in the Internet and

an implementation architecture of the proposal in open source UNIX firewalls and application end-hosts.

Chapter 1

Introduction

In the early 1980s when IPv4 was being designed, a 32 bit address space was believed to be sufficient for the Internet and the exponential growth of the Internet was definitely not foreseen. A decade later, the Internet research community started investigations for solutions to address IPv4's address space limitations. Two main solutions were proposed: development of a new Internet Protocol, IPv6, and Network Address Translators (NATs) for multiplexing private IPv4 addresses onto globally routable IPv4 addresses. During the same period network perimeter security started to be massively deployed in the Internet with its key element being Firewalls.

Although believed to be useful for the Internet, both NATs and Firewalls had unforeseen impacts; several applications that were not widely deployed when NATs and Firewalls were originally introduced were impacted. Some of these, including Voice over IP and Video over IP, are critical for todays economy and social life. Several solutions have been proposed starting the middle of the 1990s with the deployment of SOCKS and application proxies, which will allow continued usage of network perimeter security while preserving application friendliness. In the late 1990s, Application Layer Gateways (ALGs) appeared in the Internet to solve the applications' traversal of NAT and Firewalls. All these solutions had various drawbacks and led the Internet research community and the industry to investigate alternative techniques such as reflector servers, media proxies and NAT and Firewall signaling protocols. The PhD candidate was involved in all these investigations and has been actively involved in analyzing architectural options for NAT and Firewall signaling protocols which is the core of this dissertation.

1.1 Thesis goals

The goals for the thesis were to:

- Analyze NAT and Firewall impacts on the applications using the Internet and the protocols which underlie these applications
- Analyze existing solutions (as of the middle of 2002) and the usage of NAT and Firewall signaling protocols
- Define a framework for using NAT and Firewall Signaling that does not necessarily require topology knowledge by the parties involved in the signaling
- Specify a topology agnostic[1] NAT and Firewall signaling protocol
- Analyze requirements that would be imposed on the network infrastructure in deploying the proposed topology agnostic NAT and Firewall signaling protocol
- Define an implementation model of the proposed NAT and Firewall signaling protocol in a UNIX®[2] Firewall and on an application end-host

1.2 How to read this dissertation

The dissertation is split into eight chapters. Readers familiar with NAT and Firewall issues with the Internet's applications and existing traversal solutions could skip Chapter 2. Chapter 7 is interesting for people seeking to implement the proposed PDNFS protocol in a firewall or NAT product, otherwise it could be skipped. The manuscript was written in LaTeX using pdfLaTex and has embedded hyper-references which can be used to access the target sections or chapters. Furthermore every chapter includes a terminology section or refers the reader to the terminology sections in other chapters. All acronyms can be found in the index section at the end of the document with pointers to their definition. Finally the bibliography provides hyperlinks to sections referring to the bibliographic reference. For obvious privacy concerns, all network addresses used in this dissertation are private addresses as specified in [22]; hence globally

[1] By topology agnostic we mean that it does not require topology knowledge

[2] UNIX®is a registered trademark of The Open Group

routable IPv4 addresses discussed in examples are taken out of the 172.16.0.0/12 (private-use Networks address block [22]).

Chapter 2 documents a detailed analysis of NAT and Firewall impacts on the Internet's applications and provides a methodology for identifying impacted applications. Furthermore the chapter discusses existing solutions as of the start of the work described in this thesis in 2002 and discusses all the advantages and disadvantages of the available solutions, and compares them in this context. As a result of this comparison, we identify the need for NAT and Firewall signaling protocols discussed in Chapter 3

Chapter 3 discusses the architectural framework of NAT and Firewall signaling while providing a thorough comparison of signaling protocol approaches, as well as analyzing NAT and Firewall signaling protocols requirements for a global deployment of such protocols in the Internet. The conclusion of the analysis in this chapter is the necessity to use a novel two layer signaling technology: the Path-Directed Signaling Protocol (**PDSP**) technology.

Chapter 4 proposes a Path-Directed Message Transfer Protocol (**PDMTP**). This protocol is the unique component of the Common Path-Directed Signaling layer (**CPDSL**) introduced in Chapter 3. The PDMTP protocol would be used to discover neighboring nodes and transfer the messages of the protocol (and other protocols part of the PDSP framework) proposed in Chapter 5. The PDMTP protocol was heavily inspired by the Internet Engineering Task Force (**IETF**) Next Steps In Signaling (**NSIS**) Working Group's [23] General Internet Signaling Transport (**GIST**) protocol, to which the PhD candidate made significant contributions.

Chapter 5 describes the Path-Directed NAT and Firewall Signaling (**PDNFS**) protocol proposal that has been reused in the IETF's NSIS Working Group [23] to define the NAT and Firewall NSIS Signaling Layer Protocol (**NATFW NSLP**) [14]. The PhD candidate contributed significantly in the standardization efforts of the NATFW NSLP by co-authoring the protocol specification document [14]. Since there are always process delays in defining standards, the PDNFS proposal differs from the NSIS WG's specification and could be considered as more mature than the current NSIS specification.

Chapter 6 investigates the implications of deploying the proposed PDNFS protocol in the Internet, with particular emphasis on four aspects: the assessment of the impact on real time features, providing a smooth migration path from current NAT and Firewall traversal solutions,

dealing with routing asymmetry and the requirements that PDNFS imposes on the security infrastructure.

Chapter 7 provides software architecture examples for the implementation of PDNFS in a UNIX-based open source firewall (OpendBSD's PF [21]) and on application end-hosts.

We conclude in Chapter 8 and discuss the overall benefits of PDNFS and its global deployment challenges in the Internet.

1.3 Thesis publications and contributions

During the thesis the PhD candidate has made significant contributions to the IETF NSIS WG protocol specifications using the concepts researched to accomplish the goals of the thesis and published four papers.

Protocol specification contributions in the IETF:

- Contributed to the NSIS framework [16] and the NSIS threats [24] WG documents
- Contributed to the General Internet Signaling Transport specification [17] being defined in the IETF NSIS WG, this specification uses some concepts of the Path-Directed Message Transfer Protocol discussed in Chapter 4
- Co-authored the NATFW NSLP [14] specification which is an incarnation of an early version of the PDNFS protocol presented in Chapter 5 as well as several other drafts analyzing NATFW NSLP migration issues [25], security threats [26] and protocol optimization in private networks [27]
- Co-authored the NAT-PT [5] deprecation statement as part of the IPv6 Operation (V6OPS) IETF WG, this document summaries all issues relevant to Network Address Translators (the emphasis is on the IPv4-IPv6 protocol translator)
- Authored early reflections [28] [29] on path-directed signaling protocols and their comparison with other types of protocols (although we have not used that terminology at the start of our research work)

Published papers:

- *Path-Directed Signaling in the Internet* [20], discusses how to use Path-Directed Signaling protocols such as the PDNFS protocol in the Internet with or without support on Internet applications' hosts.

- *Solving asymmetric routing and load balancing issues for firewalls* [18], provides enhancements to existing firewall state synching mechanisms required for deploying the PDNFS protocol in multi-homed networks. These enhancements will be discussed in Chapter 6.

- *Interaction of Firewalls and Network Address Translators with the Internet Applications* [8], provides a summary of Chapter 2.

- *Securing Middlebox Discovery for Path-Directed Signaling in the Internet* [19], presents usage of Public Key Certificates (PKC) with specific extensions to assert a Firewall or NAT role in a specific network identified by its serviced network prefixes. These extensions will be discussed in Chapter 6.

In addition the PhD candidate made significant contributions to the IETF MIDdlebox COMmunications (MIDCOM) Working Group (WG), while analyzing NAT and Firewall traversal methods; in [30], [9], [10], [31], [32] and [33].

Chapter 2

Overview of NAT and Firewall issues with the Internet's applications

2.1 Introduction

The exponential growth of Internet users, combined with the wide deployment of (almost) permanent broadband interconnections (versus ephemeral dial-up interconnections), has introduced new addressing challenges due to the limitations of the 32 bit IPv4 addressing structure. To combat the IPv4 address depletion problem, Network Address Translators (NATs) [2] are deployed.

The same growth rate of Internet users has increased the number of attractive targets and malicious users. Hence to counter security threats that could be posed by malicious users hosted in different trust domains, network (perimeter) security tools such as Firewalls [1] are deployed.

Both Firewalls and NATs disrupt several Internet applications that are in common use:

- Firewalls:
  - Firewalls prevent the use of certain applications between hosts situated in different trust domains. Particularly affected are applications using a control signaling channel and separate data exchange streams that cannot be identified by a predetermined matching filter expression. Such applications include some that are now becoming widely used, such as Voice over IP or Video over IP which use dynamic ephemeral ports.
- NATs:

- NATs impacts all applications that carry IP addresses and ports embedded within their messages (this is the case of applications having a control signaling channel and separate data exchange streams), as well as applications that do not have keep-alive capabilities to prevent a NAT binding from being deleted during an idle period. Voice over IP (see [4, 34–36]), Video over IP and Internet gaming deployment issues have started to show the architectural burden that NATs have imposed on the Internet.

Overlay networks and application specific solutions were used in the past to solve NAT and Firewall application traversal problems. In 2002 many efforts headed by international standards communities and private consortiums were trying to document and solve these issues (see [11, 37,38]). The goals of this chapter are to introduce all these activities and their drawbacks which induced the PhD candidate to investigate alternative solutions to solve NAT and Firewall issues for real-time applications in the Internet.

2.1.1 Terminology

- Middlebox (**MB**):
  - According to [39] a middlebox is defined as any network intermediary performing data plane functions other than the basic standard functions (routing and forwarding) of an IP router on the data path between a source host and destination host. Middlebox terminology is well documented in [39] and NATs and Firewall functions are examples of functions hosted on a Middlebox (MB). In this document we use the term Middlebox to refer to any network intermediary implementing either NAT or Firewall functions.
- Filter:
  - An expression that describes the characteristics of a flow in terms of values of specific fields in the packet (one example is the classical 'five tuple': source IP address, destination IP address, transport protocol type, source port number, destination port number).
- Action:

  - Operation performed on a packet arriving at a Middlebox. A multitude of actions could be enforced on a packet flow; in this document only forward, drop, and address or address/port translation actions are considered. An optional complementary action is rate limitation which could be used in conjunction with the previous actions.

- Policy Rule:
  - The coupling of an action and one or more filter(s).

- Access Control List (**ACL**):
  - In this context, an ACL provides the list of flows that are authorized to be forwarded through a firewall or barred from passing through the firewall. An ACL is an example of a list of policy rules where the only authorized actions are drop (where the default policy on the firewall is to forward packets) or forward (where the default policy on the firewall is to drop packets).

- Realm or Address Realm:
  - An addressing domain where there is no address ambiguity (i.e., every address is unique in a realm).

- Media Proxy (**MP**):
  - A device that relays media traffic from one host to another. This device is used when application hosts behind NATs are not able to learn their translated address and port on their own.

- Mapped Address and port
  - An address and port pair selected from a pool of addresses and ports that would be used to reach an end-host system through a NAT

- NAT Binding:

  - A NAT binding provides a mapping of an address (or address and port) from a specific realm (address is hence unique in its realm) to another address (or address and port) from another address realm. In the most frequent cases, the address translation maps addresses from a private realm into addresses from the globally unique public address realm (and vice versa).

- Pinhole:
  - A pinhole is a policy rule that allows packets matching a filter expression to be forwarded by a firewall.

- **DMZ**:
  - DeMilitarized Zone, a subnetwork placed between a public and private network as a buffer zone. The DMZ is typically used as the connection point for server hosts which are intended to be accessed both from the public and private networks and has less restrictive access policies than the main private network. The DMZ is isolated from both private and public networks by firewalls so that public access can be controlled and in the event of a host in the DMZ being compromised, possibly as a result of the freer public access, the damage can be contained within the DMZ and not spread to the private network.

2.2 NATs and Firewalls Overview

2.2.1 An Introduction to Firewalls

A Firewall instance prevents certain categories of users or applications from making use of a network by filtering the packets as they pass into and out of the network through the Firewall instance.

The rules that are used to determine whether an IP packet can pass through the firewall could be based on:

- Packet Filtering:

- Based on specifying specific filters that include either values from the IP packet header and/or the port numbers in the transport protocol header using either or both source and destination parameters. Some packet filters utilize dynamically created state information that allows certain external sources to send inbound packets only if prior packets were sent to them from internal trusted hosts. The lifetime of the dynamically established state can be arbitrary and typically depends on the activity of outbound traffic being sent from trusted hosts (deployed inside the secured perimeter). The state expires if no packets are sent by trusted hosts for a determined period. Once the state is removed, inbound packets sent from sources associated with the previous valid state will be dropped. Firewalls with this capability are usually known as stateful firewalls.

- Application Layer Gateway (**ALG**):

  - This approach requires application protocol knowledge to be added to a Firewall, which is typically complex since parts of the application state machine need to be implemented on the Firewall instance. There is a coupling of application intelligence and middlebox functionality in devices hosting ALGs. These firewalls are known as 'stateful (payload or packet) inspection' firewalls. ALGs may also be associated to NATs.

In this document, we shall only consider a firewall as being either a stateless or stateful packet filter hence a firewall instance is completely decoupled from any application aware entity.

2.2.2 An Introduction to Network Address Translation

The expansion of the Internet for commercial purposes has dramatically increased the number of connected hosts, to the extent that the available globally unique IPv4 32 bit addresses are no longer sufficient. To counter the lack of globally unique IPv4 addresses, Network Address Translators have been introduced [2].

Although IPv6 deployments are starting, we do not foresee NATs disappearing for a very long time since dual stack nodes [40] (i.e. supporting both IPv4 and IPv6) would still need

to communicate with IPv4 only devices (legacy devices that can't be upgraded). Hence if dual stack application proxies are not used (and this is not an IPv6 recommended migration approach), IPv4 would still be the only networking protocol to be used by dual stack end-hosts. Therefore IPv4 NATs would still need to be deployed until all services are provided over IPv6 (and IPv4 for IPv4 only end-hosts).

There are several variants of NAT: in this document we shall focus on the most frequently deployed variant which is the traditional NAT [3]. The traditional NAT [3] or Outbound NAT creates an address binding for outbound flows (i.e., flows destined to hosts external to the private address realm). There are two sub-classes in this NAT variant:

1. Basic NAT:
    - Only the source IP address of the IP packet is replaced by a public address when the IP packet goes out to the Internet (the public destination IP address is replaced by the private one when the packet comes in from the Internet).
2. Network Address and Port Translator (NAPT):
    - The source port as well as the source IP address are replaced by public ones when the packet is sent from the private address realm. For inbound packets the destination IP address and port are replaced by the original private IP address and port pair.

In the rest of the document, the term NAT will refer to basic NAT and NAPT. There are several implementation types that differ with regards to address and port mapping creation and maintenance as well as the NAT's inherent filtering capabilities; these variants were first discussed in [9] and further refined in [41]. [9] categorizes NAT implementation types by bundling the address and port mapping mechanism with the NAT's basic filtering capabilities which leads to certain inconsistencies when categorizing NATs. We now briefly summarize the main difference between NAT implementations based on the way in which address and port mappings are created. It turns out that this is the most relevant criterion when analysing NAT traversal solutions. For further details on the other properties please refer to [41].

- External mapped address or address/port is endpoint independent:

– The NAT reuses the same mapped address and port, (X1':x1') created by outbound UDP packets sent from (X1:x1) to (Y1:y1), for subsequent sessions initiated from the same internal IP address and port (X1:x1) to any external IP address and port. In this document, we shall call this type of NAT implementation, shown in Figure 2.1, an End-Point Independent Mapping (**EPIM**) NAT. From [9]'s perspective, this is a "Cone NAT" ([9]) unless the EPIM NAT has filtering capabilities. Furthermore, as shown in Figure 2.2, any external host can send a packet to the internal host, by sending a packet to the mapped external address or address/port pair. When an EPIM NAT will only forward packets from Y1 to X1':x1' but not from a different address Y2, the NAT is called a "restricted cone NAT" in [9]. When an EPIM NAT will only forward packets from Y1:y1 to X1':x1' but not from Y1:y3, the NAT is called a "restricted port cone NAT" in [9].

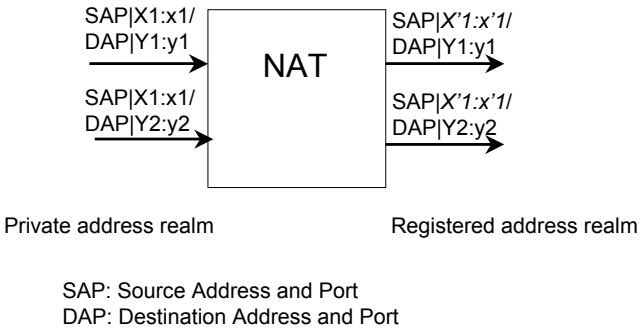


Figure 2.1: End-Point Independent Mapping NAT (EPIM)

- External mapped address (and port) is end-point address dependent:

  – The NAT reuses the port mapping for subsequent sessions initiated from the same internal IP address and port (X:x) only for sessions to the same external IP address, regardless of the external port. Specifically, X1':x1' equals X2':x2' if, and only if, Y2:y2 equals Y1:y1. In this document we shall call this NAT implementation type an End-Point Dependent Mapping (EPDM) NAT (Figure 2.3); a symmetric NAT

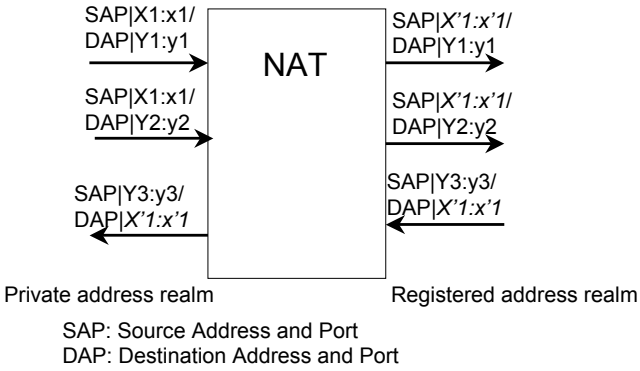


Figure 2.2: Cone NAT

as defined in [9] and shown in Figure 2.4 is an EPDM NAT with additional filtering capabilities which prevents an external host from sending packets from Y2:y2 to X'1:x1'.

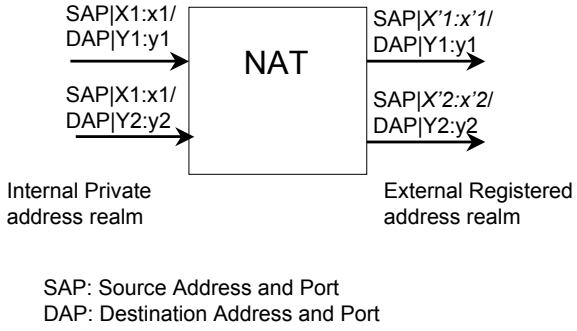


Figure 2.3: End-Point Dependent Mapping (EPDM) NAT

2.3 NAT and Firewall network deployment examples

When analyzing the impact on Internet protocols it is crucial to determine if a protocol's message may or may not traverse NATs or Firewalls, hence the requirement to analyze typical

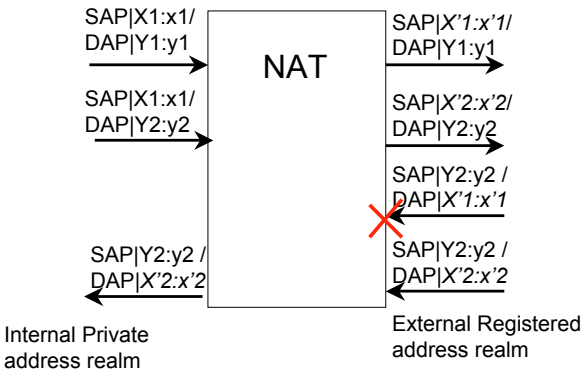


Figure 2.4: Symmetric NAT

Middlebox deployments which we will review in this section. In all network deployments host based firewalls could be deployed but are not shown in the figures.

2.3.1 Typical corporate network deployments

A typical corporate network deployment consists of one or more subnetworks protected by a firewall and a DMZ subnetwork hosting the corporate network's public servers (DNS, Email, FTP, HTTP and other servers depending on the company's business). In most cases these subnetworks are allocated private addresses which require the deployment of one or several NATs to communicate with the rest of the Internet. Figure 2.5 shows such deployments.

2.3.2 Global network deployments

Other ISP customers such as residential customers and wireless subscribers have similar networks to the ones discussed above except that these networks are simpler.
In Figure 2.6 where a number of access network types are shown, the ubiquitous deployments of Middleboxes can be seen.

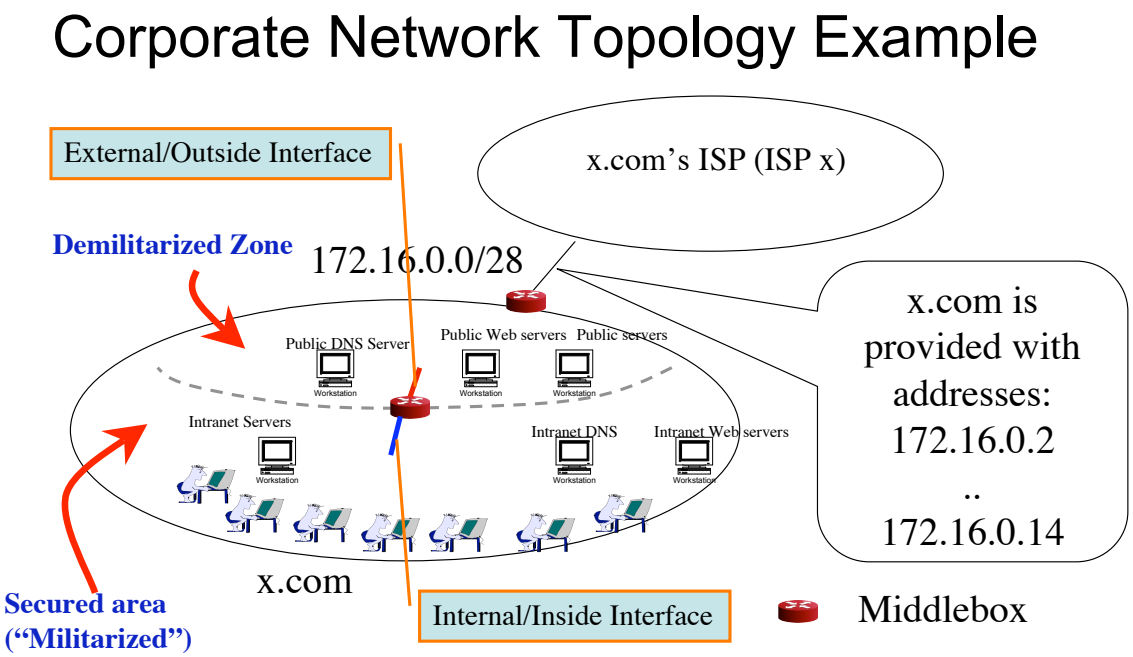


Figure 2.5: Typical corporate network deployment example

2.4 Impact of NATs and Firewalls on Internet Protocols

2.4.1 Generic categorization of Firewall impacts

Firewall impacts are mainly connectivity issues since they do not result in any packet field modification. It is impossible for one of the peers to reach its communication partner if it is not authorized to traverse the firewall (or no longer authorized in the case where pinholes are removed after a period of flow inactivity).

2.4.2 Generic categorization of NAT impacts

NAT impacts can be classified into three broad categories:

1. Connectivity: It is impossible to communicate with an entity in the private realm 'behind' the NAT as an address which we previously had is no longer valid because of expiry of

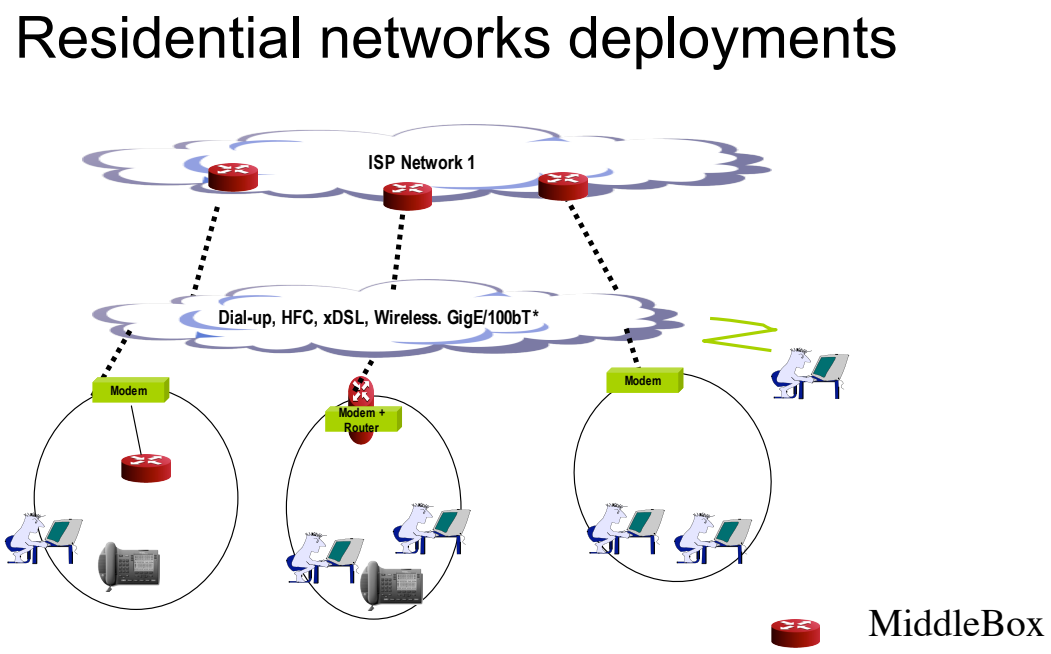


Figure 2.6: Access networks deployments

the NAT binding state; in some very specific cases packets could be prevented from being forwarded (when EPIM NATs with filtering capabilities are on the data path; this matter is discussed section 2.2.2).

2. Referrals: NAT is incompatible with protocols which have numeric IP addresses, and/or transport ports, embedded in their protocol data units that are used for referrals (i.e. the address is used or passed on by the receiver for use in making additional connections to the source). Examples of such protocols are SIP (and other VoIP or Video over IP protocols) and mobility protocols (Mobile IP, MOBIKE, etc.).

3. Security: NAT is incompatible with security mechanisms such as packet integrity checking mechanisms which assure the integrity of address and port information. When these security mechanisms are used, entities (the NAT or other devices) knowing the mapped address and ports should recompute the hashes used for integrity protection purposes;

else the integrity verification procedure will fail.

2.5 NAT and Firewall Traversal Solutions

NAT and Firewall traversal mechanisms can be grouped into four main categories:

Category 1: Concealment Solutions

Use mechanisms to bypass NATs and Firewalls; this is normally achieved by the usage of tunneling (for example, L2TP, IPsec or HTTP) mechanisms or overlay networks (for example with separate physical network interconnections or virtual ones such as VLANs or ATM Virtual Circuits). These mechanisms generally constitute security breaches (or at least policy breaches) because the actual traffic being transmitted is either concealed in the tunnel or completely bypasses the firewall, thereby subverting the rules in the Firewall. Accordingly, these mechanisms should not be encouraged especially when the two end-points are not in the same trust domain.

Category 2: Proxy Solutions

In specific cases, one could use reflector mechanisms or Media Proxies to solve application issues inherent to NATs and have tailored Firewall configurations for application flows initiated from hosts deployed in the secured network perimeter (either the policy rules in the firewall are static, or dynamic if the firewall is stateful, i.e. allow flows initiated from trusted hosts located inside the secured network perimeter).

Category 3: Application Layer Gateway (ALG) Solutions

Implement application awareness on the NAT or Firewall; this approach is commonly known as provision of an Application Level Gateway or ALG.

Category 4: Signaling Solutions

Allow application aware devices to request the NAT or the Firewall to allocate NAT binds or open pinholes in order for the application's data flows to have the desired treatment.

2.5.1 Concealment Solutions analysis

Concealment solutions have been mostly used in VPN deployments but are also used to circumvent Firewalls and NATs. Unfortunately, these solutions are also misused to bypass strict policy rules enforced by Firewalls.

The advantages are:

- No implementation or upgrade effort is needed to provide support on existing Middelboxes or on the application hosts (unless the tunneling mechanisms are implemented on the host rather than using a tunneling gateway).
- Concealment works for all applications

The disavantages are:

- Concealment may require special devices to be deployed in the network, the hardware required depends on the chosen tunneling approach (IPsec, L2TP, PPTP, etc) or overlay network (virtual interconnections such as VLANs/virtual circuits, or physical interconnections (different physical links)).
- If the solution is used to work around NATs then all the communicating hosts talking to each other must have unique IP addresses. This would be the case in a VPN.
- Since these solutions allow firewalls to be bypassed, they should only be used when the hosts communicating using these solutions, are part of the same trust domain.
- Hosts belonging to different networks should not use these mechanisms as they introduce serious security breaches: the Firewalls' existing policy rules are circumvented (tunnels are allowed to go through and generally no policies are enforced on the tunneled packets; this is also applicable to overlaid networks since firewalls are completely bypassed).

2.5.2 Proxy Solutions analysis

This category of solutions uses application aware entities to communicate with either reflector servers [9] or media proxies [42] to learn the translated addresses needed to establish communi-

cations. This category has already been identified by the IETF as a temporary expedient that people could use until signaling solutions can be deployed.

Several equipment vendors have previously implemented reflector or echo servers which respond to messages sent in a UDP datagram by the end-host, informing it of mappings that may have been made on the source address and port of the original UDP datagram. The IETF has specified such a protocol, the Simple Traversal of UDP through NATs (STUN [9]). The basic protocol idea is shown in Figure 2.7. The STUN protocol allows the mutual authentication of the end-host running the STUN client and the STUN server. The communication between the STUN client and the STUN server then creates a NAT binding which is later reused for the data traffic between the data sender and the data receiver (and likely also for traffic in the reverse direction as well).

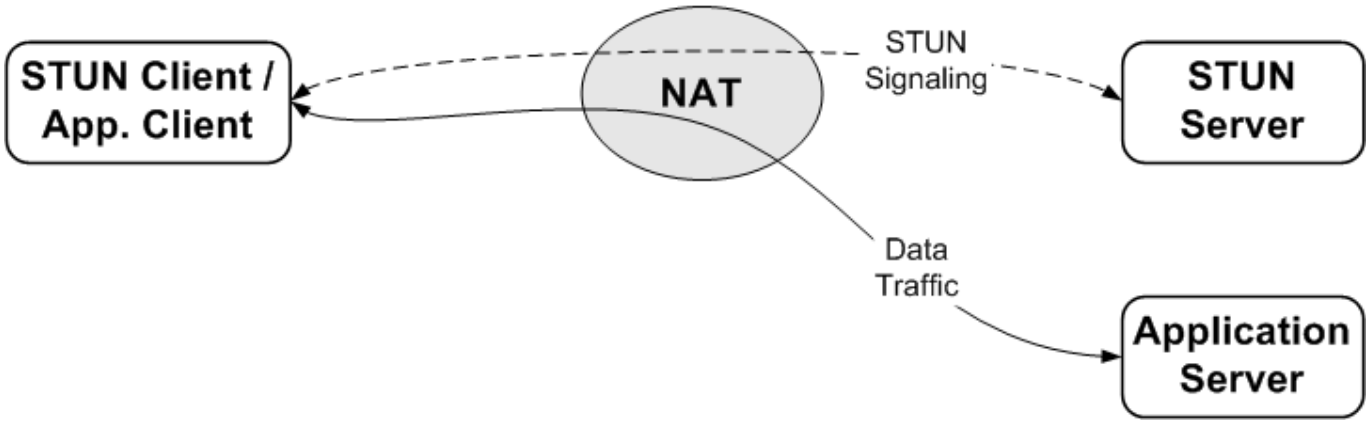


Figure 2.7: STUN basic message sequence

An alternative solution is provided by the TURN relay [42] which is a specific type of Media Proxy that is normally used in a deployment scenario where the application host runs the TURN client.

The TURN client requests the TURN relay to allocate address and port pairs where the application host will be sending its data.

The advantages for Proxy solutions are:

- No upgrade required for existing NATs
- Application signaling protocols could use cryptographic authentication and integrity protection mechanisms as well as payload encryption

The disadvantages are:

- For reflector approaches see [9]:
  - The application host needs to implement a reflector client.
  - Reflector approaches work only for UDP (i.e. not for TCP) where EPIM NATs are deployed. Reflector approaches do not work with TCP due to the NAT implementations which creates end-point dependent mappings for TCP, hence reflector approaches can't be used.
  - A reflector server needs to be deployed in the network, and in a specific way such that packets sent by the STUN client go through the NATs and then reach the STUN server.
  - The reflector message needs to traverse the same NATs as the media flows, otherwise the information which the application client will provide in the application signaling (an address and/or a port number) might not be valid during the whole application session (the NAT bind created by the STUN message and used by the external party will not be refreshed).
  - Compared to other solutions in this category, reflector approaches have issues with packet replication when used for Legal Interception purposes. If the packet replication is not enforced at the edge of the ISP it will not be transparent as in most cases the service provider terminates the traffic on a specific host serving as an intermediary. A user using a simple packet sniffer would be able to detect that he is being tapped. This problem does not occur when Media Proxies are used for all users and all application flows having NAT traversal issues.
  - Since manual policy rules need to be configured on a firewall to allow the traversal of packets, the policy rules need to allow any host to send packets inside the secured perimeter (as long as the packet flow was initiated by an internal host). Certain network administrators would prefer to prevent the deployment of an application for this reason. This problem is not applicable when Media Proxies are used because policy rules could be configured to allow communication with a finite number of

Media Proxies (vs allowing any host to send traffic as is the case with reflector solutions).

- When using Media Proxies (MP) [43]:
  - Media traffic is proxied and hence suffers from media triangle routing which may require additional bandwidth on the network infrastructure links near the MP (this is also true for TURN relays).
  - The centralized application proxy needs to be aware that the application host is behind a NAT and needs help to resolve the translated address and port pair.
  - In addition, it also needs to choose the MP which is most suitable for the application host (selection criteria could be: least loaded MP, MP that minimizes delay due to the media triangulation, or many other criteria based on network link costs).
- When using TURN [42]:
  - Media traffic is proxied and hence suffers from media triangle routing which may require additional bandwidth on the network infrastructure links near the TURN relay (this is an issue common to all types of MPs).
  - The TURN client needs to know which TURN relay to use.

For both reflector server and Media Proxy solutions, the application host needs to maintain the NAT bindings for the application signaling messages as well as the application data flows. [43] discusses how SIP NAT bindings can be kept alive, while [44] discusses MGCP NAT keep alives. In addition, Proxy solutions solve only the NAT traversal problems for applications; Firewall traversal is dependent on proper configuration of the firewalls at the risk of security exposures. These solutions do not allow the NAT devices to authenticate the end host triggering the NAT bind creation. Consequently trojan horses could trigger NAT bind creation inducing a DoS attack on address and port pools and system memory.

2.5.3 ALG Solutions analysis

This category of solution attempts to provide an autonomous network-based solution (the MB acts autonomously and the actual applications are not - or should not be - aware of its existence). In these solutions additional functionality in the Middlebox intercepts particular application signaling and installs specific policy rules derived by analysis of the signaling traffic. Optionally it might also modify the embedded addresses and port within application signaling messages.

The main advantage is that the application host does not need to be aware of any NATs or Firewalls within the network infrastructure: the Middlebox attempts to maintain the 'transparency' of the network between application hosts.

This solution has several disadvantages, as listed below:

- A major disadvantage is related to interoperability. Since the ALG needs to be aware of certain application protocol messages and state transitions (at a minimum the ones having interactions with the NAT and Firewall implementation and embedded addresses within the application's protocol messages which require dynamic changes of configured access control policy rule(s)), there is considerable opportunity for potential interoperability issues. These issues have been observed on a number of occasions in network deployments, particularly for applications having ambiguous protocol specifications or for applications having a lot of options that have embedded transport addresses or ACL issues.

- This solution interrupts the security association between application hosts and remote application entities. Many application protocols today use cryptographic mechanisms for either message integrity protection or encryption (when message confidentiality is required). This may be through the use of network layer security or application layer security. In either case, ALGs cannot be used because they are generally not informed about the cryptographic parameters (keys, security algorithms, etc) in use. If the packet is wholly or partially encrypted, the ALG will be unable to access the fields which it needs to examine or modify; if the ALG can access these fields, any modifications made will invalidate the integrity protection (keyed) hash embedded in the message.

- If several routing paths could be used, with different Middleboxes being traversed on each

path, the application signalling could traverse one Middlebox whilst the media flows could traverse a different Middelbox. In this case, ALGs do not work because the application state is not setup in or maintained by the Middlebox traversed by the media flows because the policy rules were installed on the Middlebox traversed by the application signaling.

- An ALG instance is required per protocol that has either firewall or NAT traversal issues. For example, there would be an ALG for H.323 [34], another one for SIP [4], another one for H.248/MEGACO [36], and so forth. Every time a protocol is added, a new ALG will be required, and every time a protocol is modified, the ALG will need to be updated. This leads to ever-increasing complexity, and thus significant processing resources, development resources, timeliness for deployment, etc.

- It is clear that the ALG should perform deep-packet inspection. Such granular checking at the application level entails a discernible performance penalty, which in certain implementations may impact the per packet forwarding delay.

- Due to their high software complexity, ALGs are expensive to build and maintain. It is also essential that the ALGs are kept in step with changes to the application protocols which often introduces a dependency on the Middlebox manufacturer to provide timely updates of the Middlebox functionality. Consequently ALG solutions can impede the deployment of new services.

2.5.4 Signaling Solutions analysis

As opposed to ALG solutions which are intended to be transparent to applications, signaling solutions explicitly involve applications by using signalled policy rule installation on MBs. Signaling solutions use a MB communication protocol between a MB and an application aware entity (or an entity acting on behalf of an application aware host). In this category, there could be two main variants: path-directed signaling protocols (see Fig.2.8) and targeted signaling protocols (see Fig.2.9).

The first variant uses a mode of operation where protocol messages are sent to a target destination and are intercepted by Middleboxes on the path obviating the need for topology

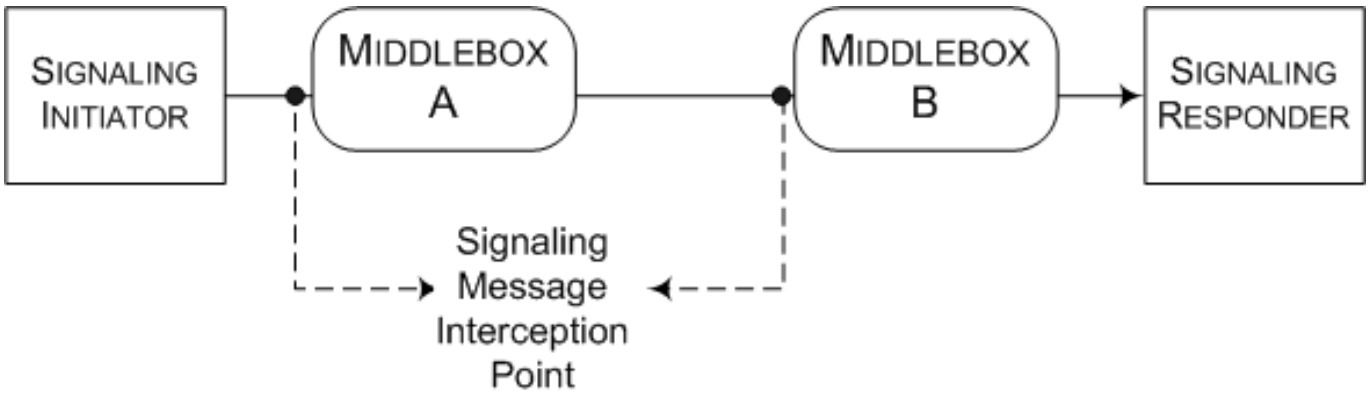


Figure 2.8: Path-directed Signaling Protocols

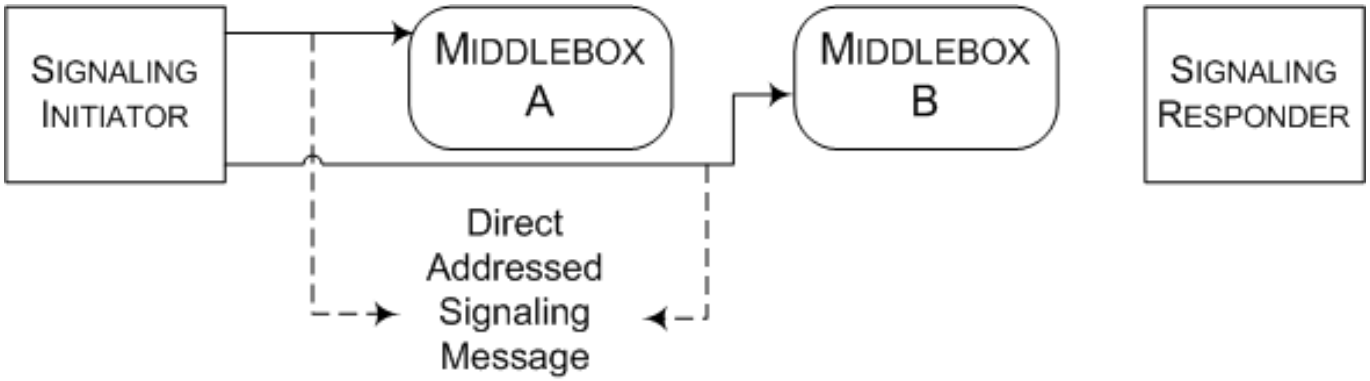


Figure 2.9: Targeted Signaling Protocols

awareness by the signaling system. In 2002 such approaches did not exist for Middleboxes, the PhD candidate has focused his work on such technology introduced in Chapter 3. As compared with path-directed signaling protocols, targeted signaling protocols require at least some minimal network topology knowledge to signal all the MBs on the data path. We shall describe further both signaling variants and their limitations in Chapter 3.

There have been three previous initiatives to develop a targeted signaling protocol, namely SOCKS [13], UPnP [11] and RSIP [12].

SOCKS was designed to only communicate with Firewalls and request the opening of pinholes.

UPnP uses a link local multicast discovery protocol to find MBs (in the same LAN segment) and a separate protocol to signal them.

RSIP was only designed to request NAT bindings, and requires the usage of a tunnel between the end-host and the NAT.

None of SOCKS, UPnP and RSIP could be considered as well-secured protocols and their use

would be inappropriate when stringent security requirements are mandated in a network environment.

From a standards activity perspective, in 2002 there were two ongoing efforts in the IETF. Next Steps in Signaling (NSIS) [23] aimed to produce a generic path-directed protocol with emphasis on the Quality Of Service signaling application, whereas MIddle Box COMunication (MIDCOM) [38] aimed for a targeted signaling protocol. In 2002 the NSIS framework allowed the signaling of various network services but was not yet chartered to handle NAT and Firewall signaling as one of the NSIS signaling applications. On the other hand, the MIDCOM protocol is NAT and Firewall signaling specific.

The Signaling solution category has the following advantages:

- Mechanisms on the MB are application independent.
- There is no need to deploy ALGs and their usual burden of software and hardware upgrades, license and operational costs.
- No application interoperability problems are encountered.
- Applications' signaling (SIP [4], H323 [34], MGCP [35] or other) messages could be encrypted without any requirement to be interpreted by the Middlebox traversed.

The solution has the following disadvantages:

- The application aware entity will need to know which MB(s) to request services from. Determining the appropriate MBs could be difficult particularly when multiple paths exist with different MBs encountered depending on the path traversed. This is an issue for solutions using targeted signaling protocols, such as MIDCOM [38], UPnP [11], RSIP [12] and SOCKS [13], which may experience problems with a large number of MBs or more complex topologies that need to be traversed by the data flows. Path-directed signaling does not suffer from this problem since the application aware entity sends its MB communication message to the far end host with which it will communicate and the MBs traversed by the message will analyze it and enforce the requested policy rules.

- Path-directed signaling assumes that when the signaling is path-coupled (a mode of signaling where the signaling messages follow a path that is tied to the data messages), no divergence occurs between the path followed by the data flows and the path of signaling messages. This might be idealistic in some scenarios but still applies to more networks as opposed to targeted signaling solutions.

- The application aware entity is no longer unaware of MBs as compared with the case where ALGs are used. The application aware entity will need to know that MBs are deployed on the path of its packet flows, as well as when to request network services from these MBs (particularly when the flows leave the administrative domain in which the application aware entity is hosted).

- Application aware entities or proxies will need to implement part of the solution (processing of MB communication messages and the required service logic) in addition to the required implementation on the MB.

- The application aware entity will need to be authenticated and authorized by the MB from which it requests service. Authenticating devices without any prior knowledge of them is possible with today's PKI technology, however it is still not the case with authorization of unknown parties. Authorizing unknown parties' (with whom a Middlebox does not have a direct trust relation) requests for Middlebox services is still a challenge.

2.6 Summary

Four solution categories to NAT and Firewall application traversal issues were discussed in the previous sections. The advantages and disadvantages of the various solution categories were compared and are also described in Table 2.1.

With the increase of network security attacks (especially trojans, viruses, worms susceptible to use weaknesses of proxy solutions and ALGs), the Internet community is seeking for secured NAT and Firewall traversal solutions that would be application independent and at the same time be extensible to handle different types of transport layer protocols. It is clear from the comparison summary shown in Table 2.1 that signaling solutions are the most appealing for the

future Internet architecture as they provide proper extensibility to handle different applications as well as different network services that extend beyond NAT and Firewall services.

Most business critical applications having NAT and Firewall issues are real-time applications. Many of these applications might also require bandwidth reservations. Hence a generic signaling solution that allows a variety (and not only NAT and Firewall signaling) of signaling applications should be the way forward for the Internet. Although generic signaling solutions are the most future proof for the Internet, until they are properly analyzed, standardized and deployed in the Internet, alternative solutions from categories 1 to 4 could be used depending on their applicability and availability.

The remainder of the manuscript will be focused on documenting a proposal for the usage of a Path-Directed NAT and Firewall signaling protocol for the Internet; that proposal would be part of a generic path-directed signaling suite for the Internet.

Table 2.1: Solutions Summary

Solutions	**Concealment Solution**	**Proxy Solution**	**ALG Solution**	**Signaling Solution**
NAT Traversal	YES	YES	YES	YES
Firewall Traversal	YES	NO	YES	YES
End-host requirements	NONE [1]	STUN/TURN only	NONE	NONE[2]
Application Impact	NONE	App. triggers STUN/TURN	NONE	App. triggers MB signaling
Middlebox requirements	NONE	NONE	YES [3]	YES [4]
Infrastructure requirements	VPN nodes	Yes[5]	NONE	AAA infrastructure
Applicability limitations	Security[6]	NAT type[7]	Security [8]	Protocol Support[9]

[1]None when VPN is provided by infrastructure

[2]Only when other entity signals the MB

[3]Support of appropriate application protocol and up-to-date on protocol versions

[4]Support of MB signaling protocol only

[5]Depends on the solution but could be the deployment of a STUN server, TURN server or Media Proxy

[6]End-host need to be in the same addressing and trust domains

[7]STUN works for UDP and only for EPIM NATs, all scenarios where firewalls are not too restrictive

[8]Works only when no cryptographic message integrity or message encryption is used

[9]Needs signaling initiator

Chapter 3

Architectural Framework for NAT and Firewall signaling in the Internet

In Chapter 2 we have compared the various NAT and Firewall traversal solutions envisaged at the start of this work and have concluded that the best solution would be to allow an application aware entity to signal requirements to Middleboxes to achieve the configuration needed to allow traversal by the application's data packets.

In this chapter we shall analyze the requirements for such Middlebox signaling solutions and propose an architectural framework for NAT and Firewall Signaling protocols.

3.1 Terminology

This chapter introduces the following new terminology:

- Path-Coupled: a mode of operation of path-directed signaling protocols (PDSPs) where the signaling protocol messages follow the same path as the data flows for which services are being signaled
- Path-Decoupled: a mode of operation of path-directed signaling protocols where the signaling protocol messages do not necessarily follow the same path as the data flows for which services are being signaled
- Resource : a file, bandwidth or any other form of asset

- Reachability Information: information required to reach a networked system, we shall use this term for information required to reach a PDSP Signaling Responder

- Signaling Initiator (**SI**): an entity, which generates a signaling message based on local or external triggers and transmits the signaling message to a specific end-host

- Signaling Responder (**SR**): an entity receiving a signaling message and responding to the signaling message originator (i.e. the Signaling Initiator). The Signaling Responder may be the end-host to which the signaling was originally directed or a proxy that responds on behalf of the end-host.

- Signaling Forwarder (**SF**): an entity that intercepts signaling messages that were not addressed to it, processes these intercepted messages and then forwards them along the path towards the Signaling Responder (or Signaling Initiator for response messages sent by the SR).

- Downstream: flow direction from the message's signaling initiator to the signaling responder

- Upstream: flow direction from the message's signaling responder to the signaling initiator

- User application signaling: signaling emanating from an application directly interacting with the end user (signaling from application layer processes). This signaling is distinguished from that generated by a signaling application such as those discussed in [45] or [16] at lower levels in the protocol stack that do not directly interact with the end-user. An example of such user application signaling is associated with Voice over IP (VoIP) applications with the signaling protocol being, for example, one of SIP [4], MGCP [35], MEGACO [36], or H323 [34].

- PDSP neighbor: a PDSP node with whom an adjacency is made at the PDSP protocol level

3.2 NAT and Firewall signaling deployment considerations

As we have seen in Chapter 2, there are various network deployment topologies for Middleboxes. We shall focus our analysis on a framework that would be applicable to any of these network deployments and is able to cope with the stages of deployment as the framework is implemented incrementally. Hence it is crucial to characterize the various possible deployments and staged deployment strategies of NAT and Firewall signaling technology.

3.2.1 Middlebox deployment characterization

There are two main network topology types for Middlebox deployments, serial (Figure 3.1) and fan(Figure 3.2); some network deployments could use a mixture of serial and fan topologies.

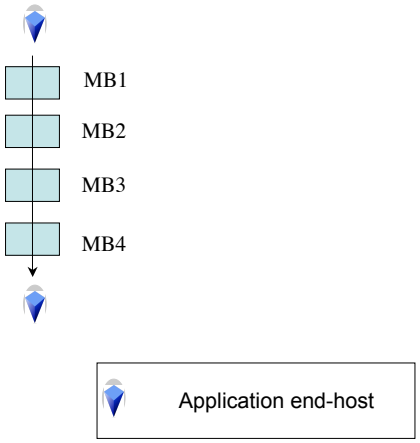


Figure 3.1: Serial Middlebox network deployment

For a serial topology there is only one path on which the Middleboxes would be traversed by packets exchanged between two end-hosts, whereas for a fan topology, Middleboxes could be deployed on several alternate paths that could be taken by packets exchanged between end-hosts.

In serial network topologies the sequence of Middlebox traversal is critical: address (or address and port) translation may be occurring in any or all of these Middleboxes. This affects the configuration needed to enable data packet flow on subsequent Middleboxes so that when signaling Middleboxes it is essential to respect this order as the signaling message related to a

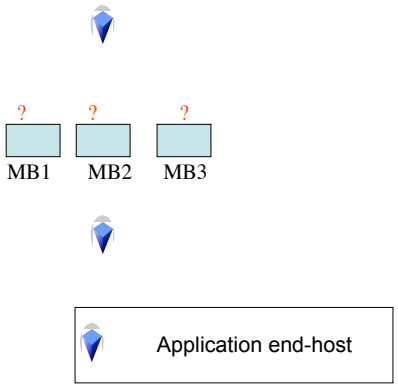


Figure 3.2: Fan Middlebox network deployment

packet flow will need to be updated to take account of changes in the packet flow description due to address (or address and port) translation in previous Middleboxes.

In fan network topologies it is difficult to predict in advance which Middlebox will be traversed by the data flows and hence it is generally not possible to determine which Middleboxes should be signaled to enable the data packet flow traversal.

3.2.2 Deployment staging and migration issues

Since the very early days of the use of IPv4 in the DARPANET it has been clear that requiring a new technology to be deployed simultaneously 'everywhere' in the Internet is undesirable. The current scale of the Internet and the requirements of continuous service in a commercially driven network make it impossible. A fundamental requirement for any new technology that expects to gain widespread acceptance in the Internet is the ability to allow incremental deployment. The ongoing migration to IPv6 shows a model for incremental deployment. Hence there are several key migration requirements for the proposed Middlebox signaling framework:

- It is not possible to predict whether Middleboxes or application end-hosts will be the first to support the Middlebox signaling protocols. Hence we need to take into account either deployment order, Middleboxes first or application end-hosts first. The Middlebox signaling framework should be robust and flexible enough to handle both cases.

- In some cases only a subset of the Middleboxes on the data path will support the proposed framework and protocol.
- In other cases either one or none of the end-hosts involved in a communication's session will support the protocol.
- The proposed Middlebox signaling framework should coexist with other existing NAT and Firewall solutions.

3.3 Signaling protocol options

As discussed in 2.5.4 we classify signaling protocols into two families, path-directed and targeted signaling protocols as shown in Figure 3.3. The main difference between the protocol families is the need for knowledge of the network topology in the signaling initiator. A path-directed protocol need not to be aware of the network topology to signal the correct Middleboxes whereas a knowledge of network topology is essential for targeted signaling.

The Path-Directed Signaling Protocol (**PDSP**) family allows a signaling initiator to send a signaling message to an end-host's address. Middleboxes (behaving as Signaling Forwarders) on the path of the signaling message will intercept the message accordingly and interpret it to provide the corresponding behavior for subsequent data flow packets implied by the message properties. They will then forward the message towards the end-host's address.

In contrast to the path-directed signaling family, the targeted signaling protocol family requires the signaling initiator to know which Middleboxes to signal and in which sequence.
The sequence of Middlebox packet traversal is critical as explained in 3.2.1, the information needed to identify the data packet flow related to the signaling message sent to a Middlebox would be dependent on the order of traversal of NATs (i.e., it needs to specify the translated address and port description that would be found in the data flow packets after they had traversed the previous NAT(s)).
In fan topologies, targeted signaling protocols are inappropriate since, as shown in Figure 3.4, it is not generally possible for a signaling initiator to determine the appropriate Middlebox to target. If the wrong Middleboxes are signaled, the data packet stream will not receive the

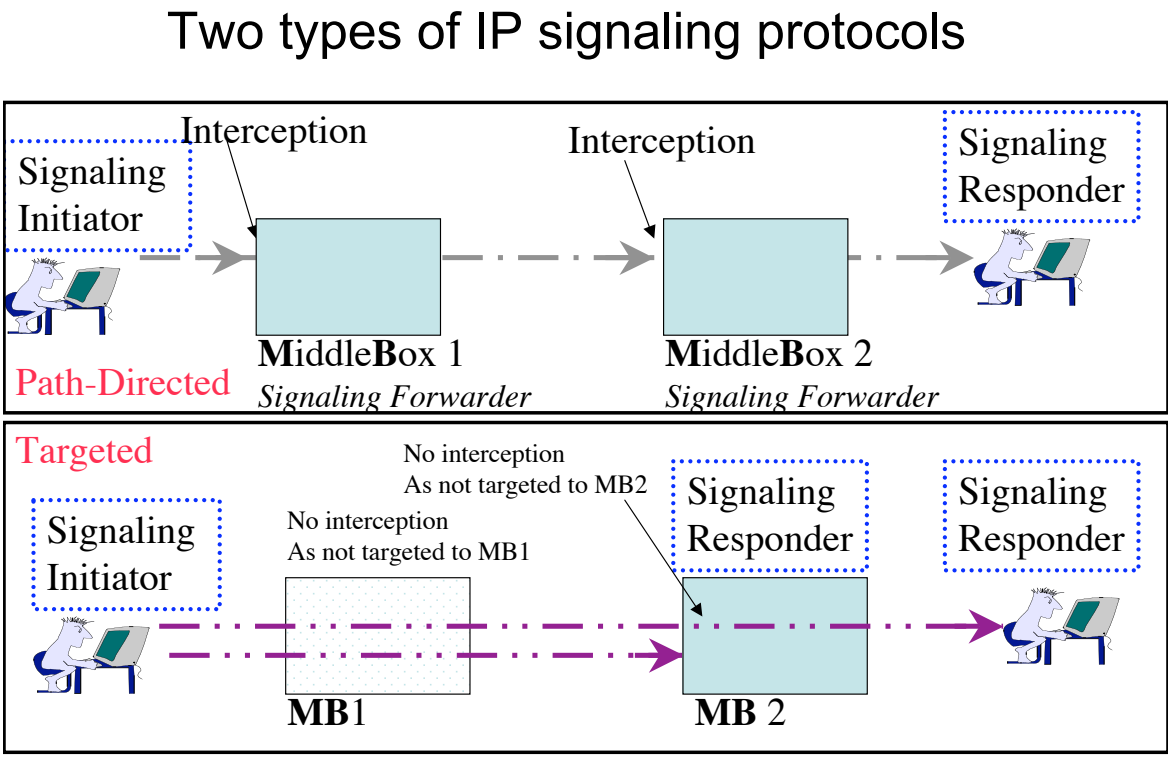


Figure 3.3: Signaling protocol families

appropriate treatment and the application will fail. With the above comparison between path-directed and targeted signaling protocol families we believe that it is safe to conclude that for a successful deployment, a NAT and Firewall signaling protocol should have the properties of a path-directed signaling protocol.

In Chapter 5 we shall discuss whether a single protocol can handle both NAT and Firewall signaling or whether two separate path-directed signaling protocols would be required to control the two classes of Middlebox.

3.3.1 Path-Directed signaling protocols' modes of operation

As discussed in 3.2.2 a Middlebox signaling framework should handle scenarios where either or both application end-hosts have not yet been upgraded to support the Middlebox signaling technology. This implies that another entity that understands the application's NAT and

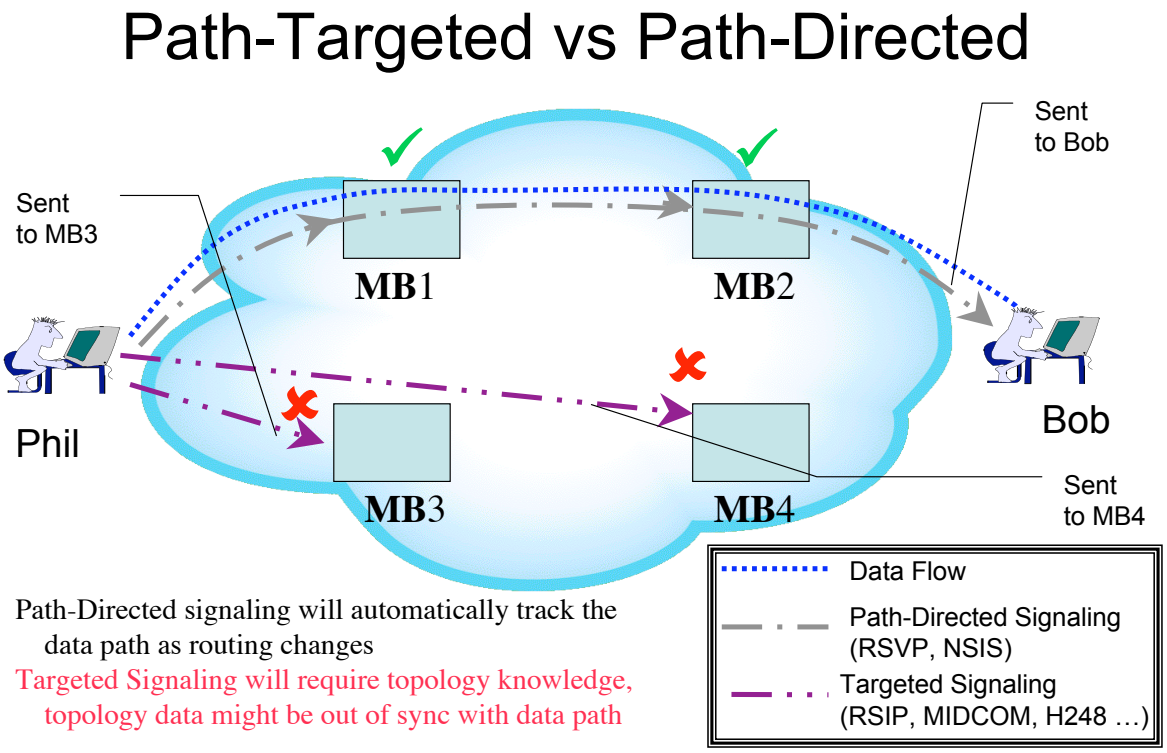


Figure 3.4: Applicability of both protocol families in fan topologies

Firewall traversal issues will need to signal the Middleboxes on the data path on behalf of an application end-host.

For this reason, in Figure 3.5, we distinguish two modes of operation for the PDSPs:

- one where the signaling protocol messages follow the same path as the data flow path, the 'path-coupled' mode of operation
- the other where the signaling protocol messages do not necessarily follow the same path as the data flow, the 'path-decoupled' mode of operation. This mode of operation is required to support network deployments where the initiating application end-host does not support the path-directed signaling protocol.

In Figure 3.6, we show a scenario where an entity trusted by the Middleboxes understands the application's requirements and sends a path-directed signaling protocol message to the application end-host (and not to the Middlebox which would be the case when a targeted signaling

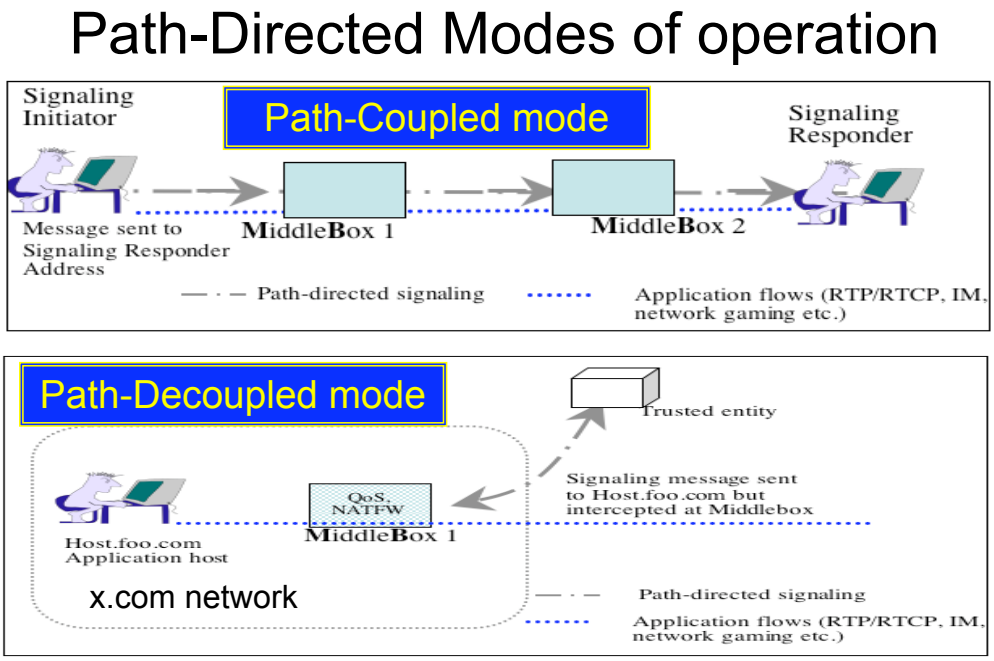


Figure 3.5: Path-Directed Signaling Protocol modes of operation

protocol is used). In Figure 3.6 the path-directed signaling messages are not routed along the hypothetical path that would have been taken by the application's flow; since the appropriate Middlebox was not dynamically configured the data flow will most likely be dropped. This is the main challenge to solve in order to allow the deployment of path-directed signaling protocols in environments where the signaling messages are not initiated by the application end-host. We shall propose in Chapter 6 a path-directed trigger signaling protocol for this purpose.

3.3.2 Path-Directed protocols: interception mechanism

The interception mechanism to be used in Signaling Forwarders is critical to path-directed signaling protocols. RSVP [46], the first standardized path-directed signaling protocol uses a IP option developed for the purpose, the Router Alert Option (RAO [47][48]). This interception mechanism was believed to be faster (on existing routers at the time when [47] was written)

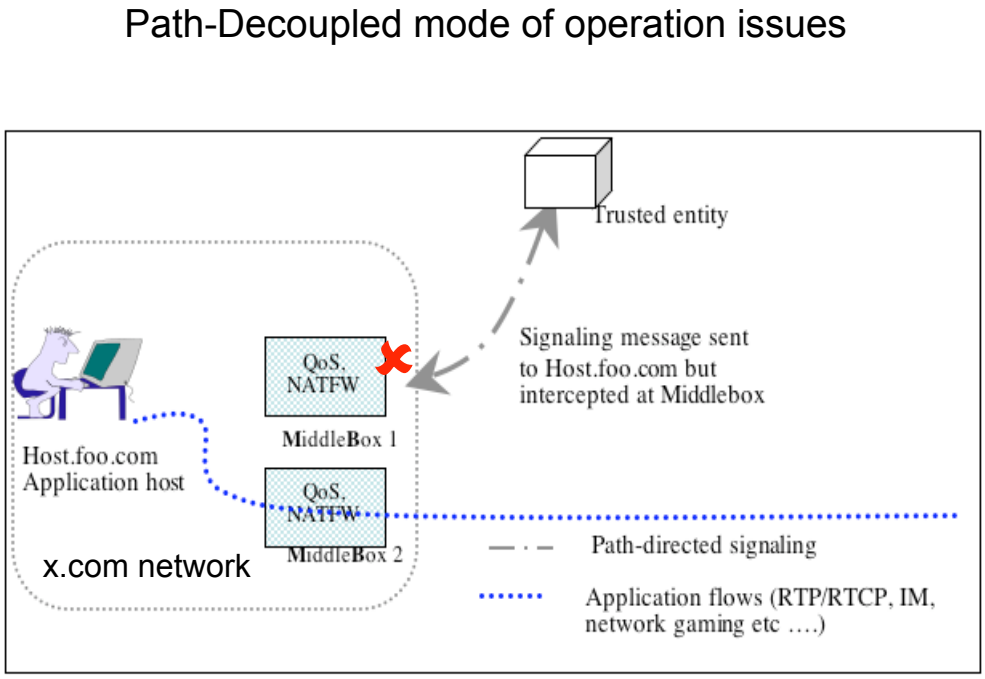


Figure 3.6: Path-Decoupled mode of operation issues

than other mechanisms (such as transparent UDP proxying, i.e., packet not addressed to the intercepting node); we assume that this is still the case. Although originally seen to be problematic on large routers handling hundreds of thousands or millions of flows due to any packet with IP options traditionally being handled on the routers' slow path, several implementations are now able to handle packets with IP options on the fast path without significant performance penalties.

In [49–51], several measurements performed in 2001, 2003 and 2004 show that packets with IP options are no longer discarded in the Internet. In addition the measured Round Trip Times of packets sent in the Internet were increased by 26% over packets without IP options. This measurement was carried out on several thousand paths, which is a relatively small proportion of the Internet but can be seen as indicative of the general behavior to be expected. Since the interception mechanism, which uses the Router Alert IP Option, is only used during the next

hop discovery phase of path-directed protocols, we believe that the use of the RAO IP option will not have a significant adverse impact on the performance of the Internet's infrastructure (particularly if path-refresh rates are low as discussed in section 3.4).

3.3.3 Path-Directed protocols: divergence from the data path

When used in the path-coupled mode, there are circumstances when path-directed protocols' messages could potentially take a different path from the data packets to which the signaling relates.

This could potentially happen when:

- Routing tables are updated after the path-directed protocol messages were sent. Given that routing tables may change after the path-directed signaling protocols' messages had installed state on Middleboxes, the path-directed protocol should have the ability to detect the resulting path changes. This could be achieved by using user application hints or other explicit detection mechanisms. Alternatively path changes could be accommodated by periodically refreshing the path with a new signaling interchange and thereby discovering the current set of downstream PDSP peers.

- In complex network topologies, routers can improve packet throughput by load balancing schemes (such as Equal Cost Multi-Path) or specific routing policies taking in account more than the packet's destination IP address to determine the next hop and spread the traffic across all the available paths. After thorough review of supported ECMP implementations [52–54], it appears that we can safely assume that most deployed routers in the Internet allocate traffic between the available paths using hashes based only on the source address and destination address (i.e., not including other possible fields such as transport port numbers or IP protocol type). Hence provided that the PDSP uses the same source and destination address for signaling messages as will be used for the prospective data flow no divergence should be expected.

The NAT and Firewall signaling framework that we shall consider in this manuscript will have all the properties necessary to detect and handle path changes. The path-change detection mechanisms are discussed in detail in section 3.4.

3.3.4 Path-Directed protocols: issues with NATs

Any NAT and Firewall signaling protocol architecture needs to consider the impact of NATs on the Internet signaling protocols as well as the inherent requirements of path-directed NAT and Firewall signaling protocols.

Based on the analysis done in 2.4.2, NATs appear to impact all path-directed signaling protocols in the following manner:

- After processing by NATs, any addressing information relating to the message source that is embedded in PDSP messages will be invalidated and will no longer be relevant to the messages' recipient (PDSP Signaling Responder or PDSP Signaling Forwarder) as the embedded address information no longer matches the translated (source) address of the data flow packets. The behavior requested by the signaling message will not be forthcoming as no packets will match the filter expressed by the embedded address information. This implies that when a path-directed signaling message with embedded addressing information traverses a NAT, the embedded addressing information should be updated to reflect the translation which the NAT will make on the data packets.

- The PDSP messages are sent from a signaling initiator to a signaling responder. The signaling initiator must therefore have a reachable address for the signaling responder before it can send useful signaling messages. However the PDSP signaling responder may be in a network connected to the Internet through a NAT. This NAT 'hides' the PDSP signaling responder so that the signaling initiator cannot acquire a reachable address unaided. Reachable address(es) for such PDSP signaling responders would only be made available by using rendez-vous server(s), where the recipients' reachability information is either provided implicitly (if the rendez-vous server analyzes received packets sent from the PDSP signaling responder and stores the translated source address and transport port) or explicitly (the signaling responder node learns for itself the translation applied by the NAT and provides the information to the rendez-vous server).

3.3.4.1 Determining the PDSP signaling responder's rendez-vous address

When networked entities are hosted behind a NAT, a rendez-vous server must be used to either learn or advertise the reachability of the 'NATed' entity if they are to act as signaling responders or data receivers for flows initiated from the other side of the NAT. In most cases a rendez-vous server is an application proxy that would be used to communicate to the user application co-hosted with the PDSP signaling responder. User application clients (for example a SIP UA [4] client) would communicate with the SIP proxy (serving as a rendez-vous server) to learn the PDSP signaling responder's reachability information. The reachability information gathered by the rendez-vous server would be provided by one of the following methods:

- Implicit (Figure 3.7): the rendez-vous server analyzes 'rendez-vous' protocol messages, most likely to be user application messages, and stores the PDSP signaling responder's rendez-vous information as the source address and source transport port of the datagram or segment that brought the rendez-vous protocol message. The server could either inform the PDSP signaling responder of its rendez-vous information or inform other entites (this would potentially be the case when the rendez-vous server is a SIP server, if the appropriate changes to the SIP protocol [4] are made).

  The STUN protocol [9] discussed in Chapter 2 could be also be considered as a rendez-vous protocol. As opposed to the SIP use case just described, STUN provides an example of explicit learning by the responder. The STUN server, situated outside the 'NATed' network, informs the STUN client co-hosted with the PDSP signaling responder about the translated address seen on packets directed to the STUN server which will also be used for its data flow (the STUN protocol uses the same communications end-point as the data flow).

- Explicit(Figure 3.8): the PDSP signaling responder communicates with all the NATs in its network and learns the appropriate translated address(es) and transport port(s) on which either PDSP signaling messages or the data flow packets will be received (the latter is preferable to minimize impacts on applications)[1]

[1]User application protocols that embed the data flow recipient's IP address and transport port number would need to be extended to support the PDSP recipient address. This extension could be avoided by just using the

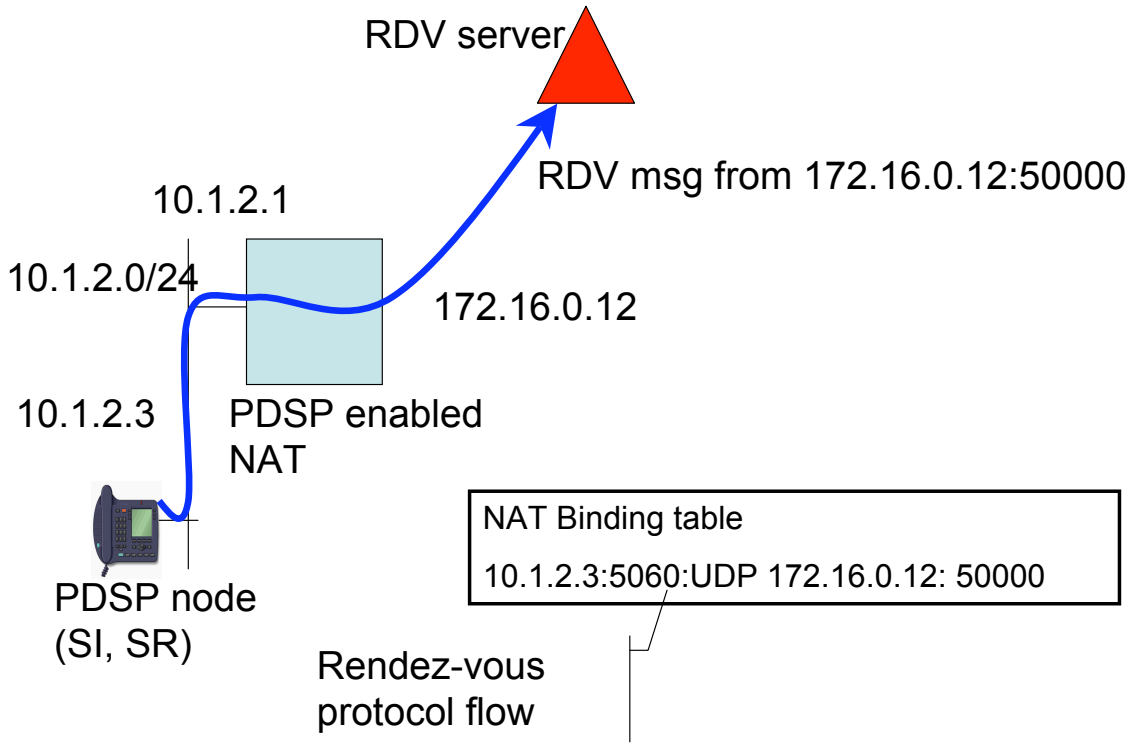


Figure 3.7: Implicit rendez-vous method

In Figures 3.7 and 3.8 we show a network hosting a PDSP enabled NAT. The example network is allocated a registered IP address (172.16.0.12) assigned to the NAT, and a private address prefix (10.1.2.0/24) servicing several nodes for which the NAT is the default gateway or on-link router providing connectivity to the Internet beyond the NAT. In Figure 3.7 the PDSP SR's reachability information (i.e. the address and port on which SIs could reach it) is determined without any action needing to be performed by the PDSP SR.

The rendez-vous protocol message sent by the rendez-vous client co-hosted with the PDSP SR allows the rendez-vous server to learn the SR's reachability or rendez-vous information. If the rendez-vous information semantics do not match the semantics of the existing user application protocols these protocols would be impacted and standardization effort would be required; other drawbacks of the approach are application protocols' backward compatibility issues. This

data flow's recipient's translated address and transport port as the rendez-vous information

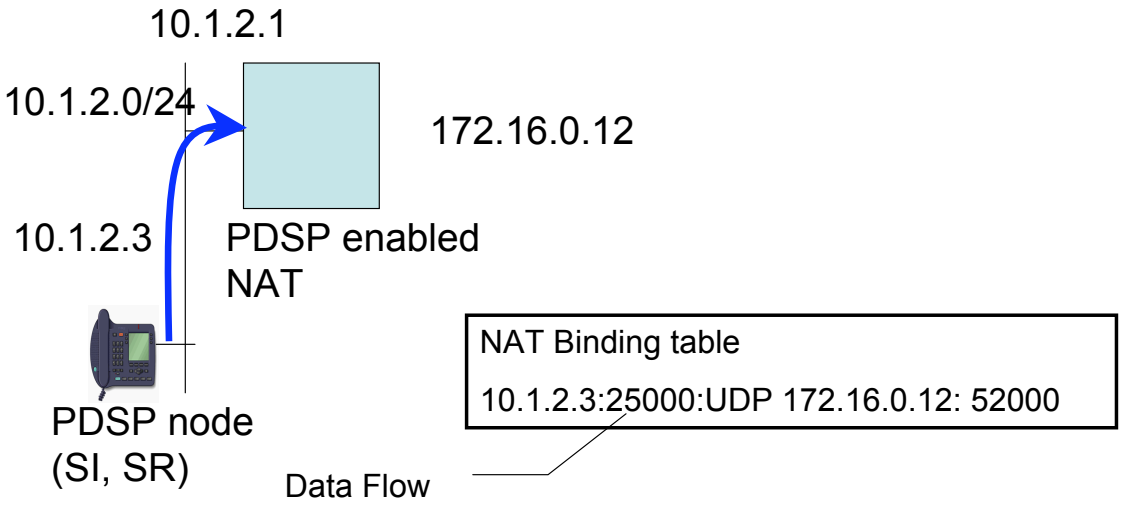


Figure 3.8: Explicit rendez-vous method

is typically the case when the rendez-vous information is tied to the rendez-vous signaling reachability information (in the example in figure 3.7 this is 172.16.0.12:50000:UDP which is derived from the NAT bind table entry; this allows the NAT to redirect the PDSP message to the PDSP SR's receiving socket end-point) and not the data flow's receiving socket end-point mapped address and port (shown in figure 3.8, 172.16.0.12:52000:UDP). Although both rendez-vous entries allow the NAT to forward the PDSP signaling message to the right SR (10.1.2.3), the entry that is not related to the data flow cannot not be used as a result of the user application impacts. We choose to make use of the explicit rendez-vous method in the protocol proposed in the remainder of this manuscript as it does not suffer from user application impacts and does not introduce potential denial of service (DoS) attacks (as would be the case with STUN as discussed in section 2.5.2) on the NAT's binding creation process.

3.3.5 Integration within existing security infrastructures

As discussed in Chapter 2, a NAT and Firewall signaling protocol needs to be protected from malicious use and should not allow new attacks on the network infrastructure. For every request made to a NAT or Firewall, proper authentication and authorization of the requesting party is required, the protocol adopted should be sufficiently flexible to allow the usage of existing authentication and authorization schemes already deployed in the Internet.
The next subsections discuss the authorization considerations relevant to our proposed NAT and Firewall path-directed protocol.

3.3.5.1 Authorization model support

We can distinguish two main authorization models in the Internet:

- Direct authorization model: requires that every entity wishing to access a resource needs to perform the necessary authorization procedures with the entity authorizing access (the Middlebox in this work) to the resource

- Transitive authorization model: requires that every entity wishing to access a resource performs the necessary authorization procedures with a trusted third party of the entity (the Middlebox) authorizing access to the resource

The first method would require that the information used during the authorization process be propagated up to the Middlebox, whereas the second method is more flexible as the Signaling Initiator could be authorized at the closest Middlebox and the other Middleboxes in the same infrastructure would implicitly authorize it.

3.3.5.2 Credential types used for the Middlebox signaling authorization process

Authorizing an entity to request the instantiation or modification of a behavior for a packet flow, requires that the entity provides the right credential matching the requested operation.
Our proposed path-directed NAT and Firewall signaling will be able to use the following credential types:

- Identity credentials: since the NAT and Firewall path-directed signaling will traverse several networks, the identity needs to be unique and accepted by all the Middleboxes traversed. Currently only the Public Key Infrastructure with X.509 certificates [55] (leaving aside host based identity frameworks such as HIP [56] that are not (yet) globally deployed) could allow an entity to authenticate itself while providing such global identity. Other local domain identity schemes could be used, such as the Kerberos principal name [57], for authentication over more restricted areas than the whole Internet. We note that, although Kerberos provides for cross-realm user authentication procedures, these procedures were never deployed, hence the local scope of Kerberos identities.
  The identity would be matched against the identity's holder privileges which would include the type of services it could perform on data flows.

- Role-based credentials: a simple authorization model would consist of authorizing entities based on their asserted role(s). The proof of authorization for the assertion to request services could be in the form of an authorization token such as the one discussed in [58] or based on the X.509 Privilege Management Infrastructure (PMI) [55].

3.3.6 Stale state handling on Middleboxes

There are two types of potential stale state that may arise when using PDSP protocols:

- PDSP Message routing and PDSP neighbor information: since the PDSP protocols use a PDSP neighbor discovery mechanism to identify the adjacent SI, SF or SR upstream or downstream along the path. Thereafter each PDSP node keeps track of the neighbor PDSP nodes that handle the messaging associated with each specific data flow. Since a PDSP node could be removed from the path due to routing changes, to avoid storage of stale message routing data (i.e., to which PDSP neighbor node should a PDSP message be sent), all message routing information kept on the PDSP node should be periodically refreshed. The refresh rate of message routing information is tightly coupled to the path-change detection mechanism in use; if the detection mechanism is accurate and fast enough the message routing state refresh rate could be very low, whereas it may need to be faster if exchange of PDSP signaling messages is used to validate the message routing

state. The state associated with message routing and PDSP neighbor knowledge will be called Message Routing State (MRS) with an associated refresh period and timer called the Message Routing State refresh period/timer. The refresh period for MRS should be related to the reliability of the network and the acceptable duration of temporary breakages in data flows.

- Behavior requested by the PDSP signaling messages: every PDSP protocol requests a certain behavior to be enforced at participating Middleboxes (in the case of the NAT and Firewall PDSP, the requested behavior would typically be either a NAT binding or a firewall pinhole opening for a data flow). The state associated with the requested behavior is linked to the PDSP SI. If the PDSP SI does not explicitly remove the requested behavior from Middleboxes when it is no longer needed for the application data flow, the state would be stored for ever or until manually removed, hence these signaled behavior states should have a finite lifetime and be automatically removed if not refreshed by the PDSP SI. In addition if a PDSP is no longer on the data path, these signaled behavior states should be removed from the Middleboxes traversed. The state associated with the data flow behavior requested by the SI will be called the Signaled Behavior State (**SBS**) with an associated refresh period and timer called the Signaled Behavior State refresh period/timer. The refresh period for SBS should be related to the expected lifetime of the associated data flows.

The treatment of all Middlebox state as 'soft state' that is deleted after a period if not refreshed explicitly is an essential requirement for a PDSP. PDSPs cannot use a 'hard state' which needs to be removed by the SI because a change of routing might remove the Middlebox from the path for a particular session. Since the SI is not explicitly aware of which Middleboxes installed state as a result of a PDSP signaling message, it has no means to signal outside the path to delete this 'orphaned' state.

The use of hard state would inevitably lead to gradual accumulation of redundant state on Middleboxes eventually resulting in resource exhaustion unless the state is removed by local administrative action which negates the purpose of the PDSP. For the user applications to be

as effective as possible, and implicitly the user experience to be as efficient as possible, it is vital that if a new Middlebox is inserted on the path the new Middlebox allows traversal of the application packets as soon as possible and preferably this should occur without a significant message routing state refresh overhead. We shall discuss in section 3.4 the possible path-change detection methods that could be used. If fast (and accurate) path-change detection methods are available, the Message Routing State refresh period could potentially be equal to the Signaled Behavior State refresh period allowing the use of a single combined timer.

3.3.7 Protocol Extensibility

The Internet's usage is continuously evolving. As a result existing Internet protocols will evolve and new protocols will emerge. Consequently NATs and Firewalls will need to be signaled to allow data flows of new protocols to traverse them.
We believe that the protocol architecture that we are proposing will provide the necessary flexibility mechanisms to describe packet flows that might be required in future.

3.4 Path-change detection

Unexpected network connection failures and predictable triggers (such as routing metric changes or introduction/removal of routers for maintenance or network upgrade) will impact the Forwarding Information Base (FIB) tables in routers, leading to paths being rerouted. Accordingly every Message Routing State entry (presented above) should have a lifetime.

There are several ways of detecting that a refresh is required for a data flow path and the associated message routing state entries:

- Detection mechanisms based on non-arrival of data flow packets at either the data sender or data receiver:
  - User application detection mechanisms at the SI:
    an example of such detection mechanism in the case of VoIP or Video over IP is the use of RTCP [59] receiver reports to detect packet losses. If the path has changed and, as a result, a Firewall or NAT (that did not have the correct SBS) appeared

on the path then all packets could be dropped; the RTCP receiver report would not be received and would be a sign that, for example, the new Firewall or NAT had appeared on the path. The same method could be used with received RTP [59] packets if the RTP packets are sent and received from the same UDP socket. Most RTP and RTCP implementations support the use of RTP and RTCP reports to detect application inactivity issues. Several well known RTP/RTCP implementations use sender and receiver reports with periods of 5s (for RTCP reports) or 25ms (for RTP) with a multiple of 5 consecutive lost reports used to detect application activity loss. Since such mechanisms are already implemented, our proposal is to reuse data packet loss reports on the end-host as means to trigger path-refresh as this would not require significant changes to the end-system applications.

– User application detection mechanisms at the SR:

  Using the same example as for SI user application detection, a similar mechanism could be employed, except that the RTP packets and the RTCP sender report would not be received, as a result the PDSP SR node will notify the PDSP SI node that the path has changed. Note that PDSP paths are unidirectional so that loss of connectivity from SI to SR does not necessarily preclude sending a notification from SR to SI.

– Unacknowledged transport protocol PDUs (TCP segments or SCTP streams): several user application protocols having NAT or Firewall issues use connection oriented transport protocols for their transport reliability. An example of such protocols, is the T.120 [60] protocol suite (which uses TCP in most implementations) used for white boarding, application sharing and file sharing. For either TCP or SCTP, for every transmitted segment or stream chunk an acknowledgment must be sent by the data receiver to the data sender; we propose to trigger the path-refresh in the event of one or two unacknowledged retransmissions.

  The detection period in this case would be dependent on the exponential backoff retransmission timer values used for the transport protocol. In most cases these parameters are inaccessible within the operation system's kernel and cannot be changed

by user applications. For example on FreeBSD[61] the TCP client retransmits unacknowledged segments after a 3s timeout, so if the PDSP SI is triggered after 1 or 2 retransmissions the overall detection time would be 6 or 9 seconds which is quite acceptable for many applications.

- Forwarding Information Base has changed on the router: although PDSP nodes are not necessarily connected physically and, for example, adjacent SFs could be several IP hops apart, a PDSP node could interpret a routing table change as a trigger to refresh the message routing state. This mechanism is not usually very accurate and may consume processor cycles unnecessarily especially if the detection is very coarse grained, resulting in all message routing entries on a node being flagged as in need of a refresh.

- Variation of the IP hop count for data flow packets relative to that of PDSP packets: if data flow and PDSP packets are taking the same path the IP hop count for the data flow should be the same as for the PDSP signaling messages used to establish the SBS. This can be measured by inspecting the TTL (Time To Live) IPv4 field or IPv6 hop count of the packets on arrival at the SR. A change in the TTL value of the data packets indicates that the path has changed for that flow. It does not imply that all message routing entries on the PDSP node need to be refreshed but only the entry associated with the data flow showing a difference. This detection mechanism might not be very accurate as the TTL variance might not always occur when the path has changed, as is clearly shown in Figure 3.9.

  This detection technique assumes that every application (user level or PDSP) mandates the use of a single initial TTL when generating the packets for a specific flow (or signaling messages related to a specific flow); this is currently left to IP stack implementors as it is not specified in [62][2]. In addition this detection method can only work if the new PDSP nodes are not dropping the data flow packets (when the PDSP node is a Firewall all the data packets would normally be dropped, for a Middlebox supporting NAT packets would only be dropped in one direction) hence this technique should not be used due to the wide deployments of NAT and Firewall MBs in the Internet. This mechanism as well as the previous might raise false Middlebox insertion/removal notifications as the same

Middleboxes would still be on the data path while new routers are inserted in the data path.

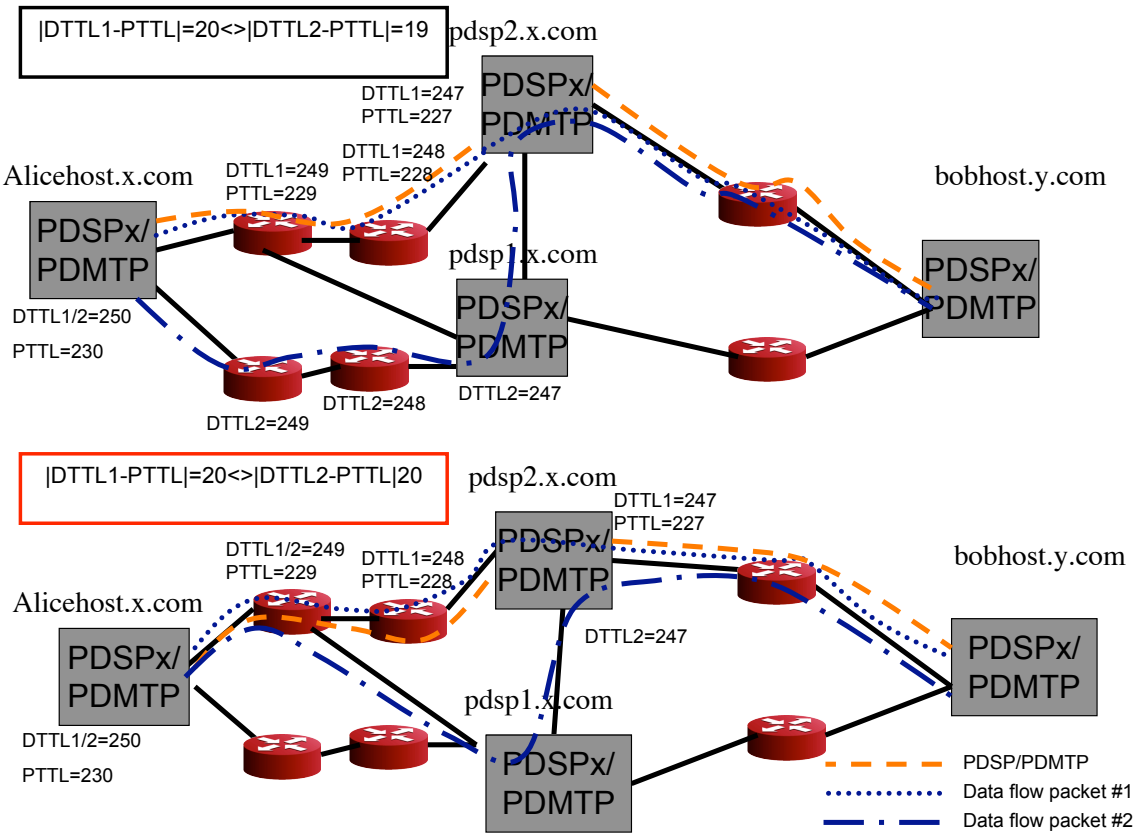


Figure 3.9: Issues with the TTL variance path-change detection technique

- No more data packets are traversing the PDSP node: this method of detection is only applicable to a downstream node that is no longer on the data flow path. A node that is still on the path cannot detect if its downstream node is unchanged but this method can be used to notify the PDSP signaling initiator that the path has changed.

Among the non SI or SR detection methods, this detection method is worthy to be implemented (as opposed to the TTL variation method and the routing table change

[2]DHCP option 23 allows automatic configuration of the IP TTL to use on a host but is not commonly deployed on IP hosts. Certain UNIX operating systems (FreeBSD, Solaris® [63], AIX® [64]) allow configuration of the IP TTL parameter with administrator privileges without requiring kernel compilation. On IPv6 networks, the equivalent default Hop Count parameter is determined by on-link routers.

method) when the PDSP supports path-change notification messages sent to the SI. This path-change notification might not be possible if the node has left the network due to mobility reasons but in that case the SI node would know of its mobility context and would trigger the path-refresh.

Mechanisms using data flow monitoring techniques on the end-host transmitting the data flows do not require any path-directed messages to notify the NAT and Firewall signaling protocol suite. For the other methods, the PDSP protocol entity that resides on the node detecting the path-change requires PDSP support to notify the PDSP Signaling Initiator about the path-change, this would ensure that a global path-refresh is performed. Since the PDSP layer is solicited to notify the PDSP Signaling Initiator, the PDSP protocol semantics need to consider this requirement.

3.5 A common layer for path-directed signaling protocols

There are several interesting applications that would benefit from the properties of Path-Directed signaling protocols. We have focused our discussion on NAT and Firewall signaling but there are several other signaling applications:

- Quality of Service (QoS) signaling
- Metering engine signaling
- Trigger signaling (discussed in Chapter 6)
- Geographic Localization signaling

All these signaling protocols have the following common requirements due to their Path-Directed nature:

- Message interception capabilities when messages are not addressed to the intermediate node receiving the message (used for discovering downstream PDSP nodes)
- Path refresh capabilities: this is needed to ensure that the right intermediate nodes (i.e. the ones on the data path) are being signaled during the lifetime of the user application session.

- Secure and reliable message delivery [3]
- Congestion control [3]
- Ordered delivery [3]
- Message bundling
- Ability to operate in both path-coupled and path-decoupled modes [3]
- Embedded addressing information [4]

In addition it is quite probable that the same networking equipment will be used for several functions that might all be signaled, hence for ease of deployment and implementation, having a common layer delivering all the commonly required functions by the PDSP protocols would significantly help their deployment.
We have therefore adopted a two layer signaling approach for PDSP protocols, initially inspired from [45], where a Common Path-Directed Signaling Layer (**CPDSL**) provides its services to the Path-Directed Signaling Application Protocol (**PDSAP**) layer.

The protocol stack, shown in Figure 3.10, of our proposed path-directed NAT and Firewall signaling protocol framework would be split into two parts:

- The CPDSL
- The signaling application layer which uses the proposed PDSAP, Path-Directed NAT and Firewall Signaling (**PDNFS**) protocol, discussed in Chapter 5

In Figure 3.10 we show a high level view of a PDNFS implementation on an Internet node that might be either a Middlebox or an end-host hosting a user level application, for example SIP [4], H323 [34], or MGCP [35].

[3] Certain protocol implementations or networks might not need this capability

[4] This might not apply to all signaling application protocols. This is applicable if the Middlebox traversed runs a NAT engine in which case the embedded addressing information should be translated. A signaling application agnostic method should be used to avoid a node supporting a NAT engine having to understand all signaling applications

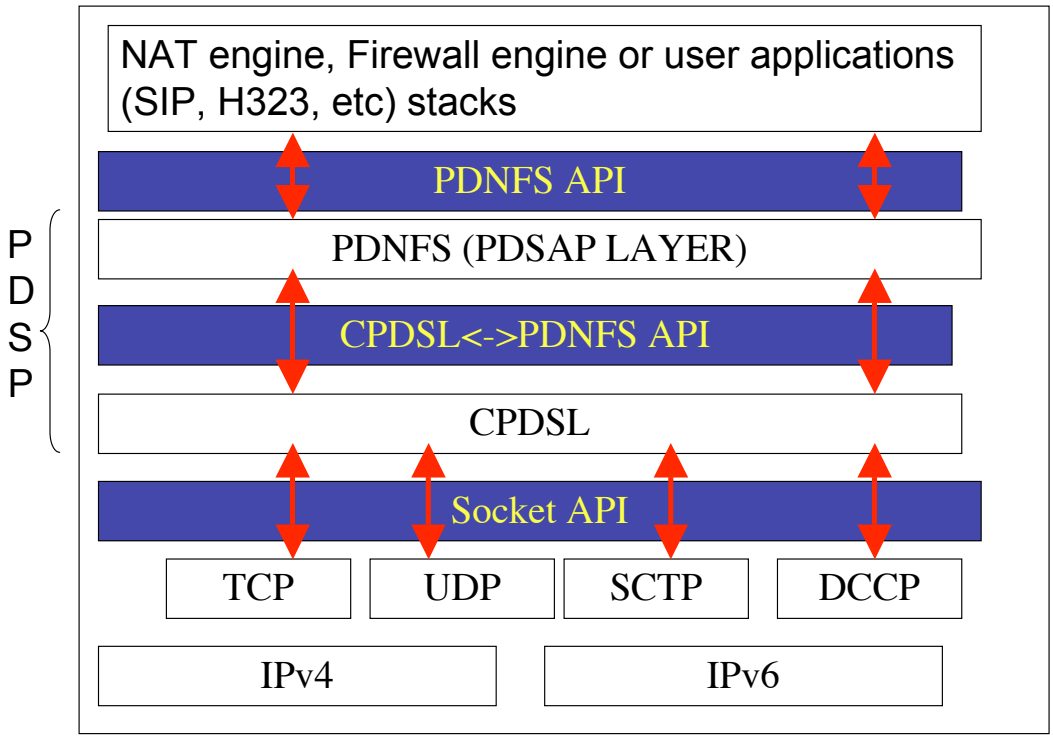


Figure 3.10: PDNFS integration on a networked system (end-host or Middlebox)

Since the Common Path-Directed Signaling Layer consists of several functional capabilities, a number of variants of the Common Path-Directed Signaling Layers can be envisaged depending on how each of these capabilities is implemented.

The following analysis is intended to identify the most useful CPDSL variants. We shall focus on the various options for the path-refresh function which is a key differentiator. Path-refresh could be handled either in a hop-by-hop fashion or end-to-end fashion as we shall see in the following subsections.

3.5.1 State refresh options for the NAT and Firewall PDSP protocol

As we discussed in section 3.3.6, there are several pieces of state with potentially independent refresh periods that need to be refreshed on PDSP nodes and that are inherent to the PDSP protocols (Message Routing State (MRS) and Signaled Behavior State (SBS)). Since an accu-

mulation of stale state could prevent PDSP nodes from accepting new messages due to lack of resources, the refreshing of potentially stale state needs to be subject to authorization. The simplest authorization model that we could use is a role-based authorization model for state refreshes where only the entity that has initially created the state could refresh it. That approach requires the usage of a proof of ownership mechanism at the layer which carries out the state refreshing.

State refresh methods use either an end-to-end refresh or a hop-by-hop refresh.

We can distinguish two types of refresh periods for both methods, the state refresh period and the refresh message (the one(s) used to refresh the state) period. The refresh message period is calculated (this is specific to the used refresh method) based on the state refresh period and other factors such as packet loss.

The advantages and disadvantages of both methods are discussed below and summarized in Figure 3.11:

- The main advantage of end-to-end refresh is that it is less processing intensive on the SF since the SFs role is only to reset the state timer or remove the state if no refresh occurred before the state timer expiry.

  However with end-to-end refresh, the required refresh message rate needed to prevent the state expiring on the SFs cannot be determined in an obvious way. The relation between the two rates (MRS refresh rate and the refresh message rate) depends on expected packet loss, per SF hop additional processing delay and propagation time to reach the SF (this is not expected to be a problem as we expect that state refresh timers would have values in the order of several seconds).

  To avoid setting high refresh message rates, we propose to use a feedback mechanism where the first PDSP node having issues with the refresh message rate, notifies the PDSP SI that it considers the refresh message rate to be too low. This would allow the PDSP SI to start with a small refresh message rate (for example 2 times greater than the state refresh rate) and increase the refresh message rate by a certain multiplying factor (for example 2 or 1.5) when a node indicates that the refresh message rate should be extended.

- The main advantage of hop-by-hop refresh, is that every peer pair decides on the state

refresh period, since the refresh is hop-by-hop the refresh message rate is simpler to determine (only one PDSP hop, no need to integrate retransmissions to go beyond one PDSP hop).

However this refresh method is expected to require more processing resources on the PDSP SFs as every PDSP SF needs to set a timer for refresh messages incoming from its upstream PDSP peer and if a message is received before the timer expiry (negotiated with the upstream peer) send a refresh message downstream when its own state timer expires.

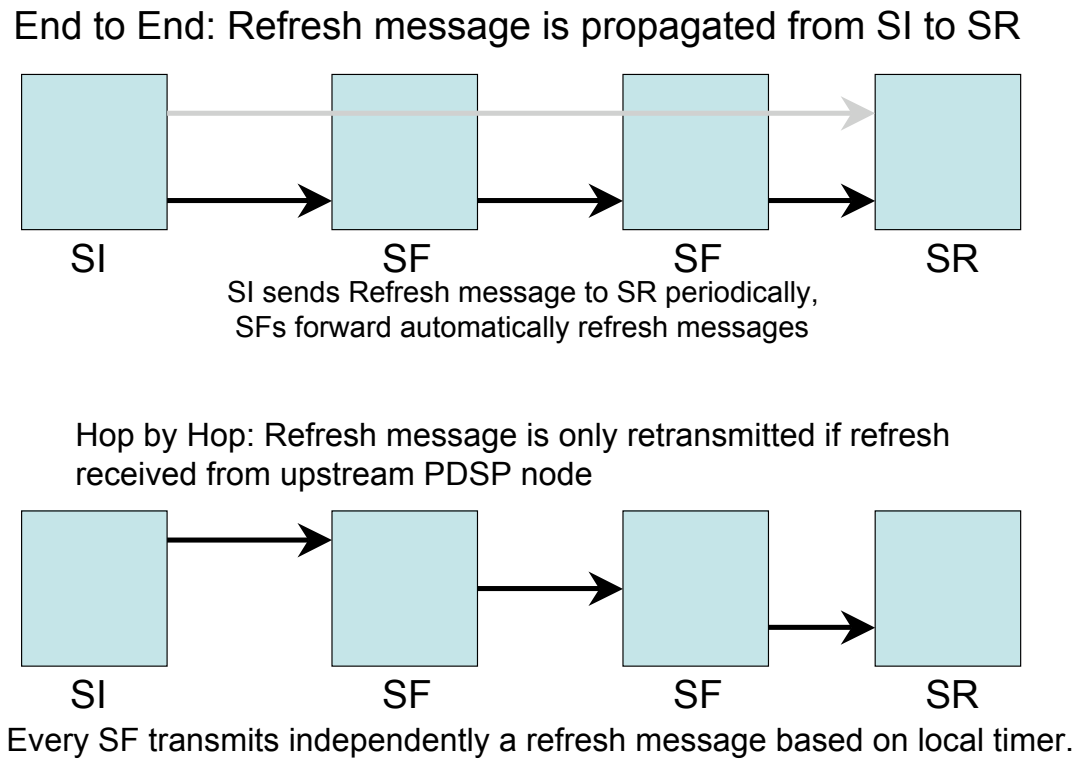


Figure 3.11: End-to-end refresh vs hop-by-hop refresh

3.5.1.1 Message routing state refresh mechanisms

Since Message Routing State (MRS) should be refreshed frequently (especially if fast and accurate path-change detection mechanisms are not available), in the absence of a feedback

mechanism to set the proper end-to-end refresh message rate, end-to-end refresh would not be possible without a large refresh message overhead.

Hence to ensure that refresh messages are reaching every SF before the MRS timer expires, the refresh message rate would deliver three to four refresh messages (or more) during a MRS timer period.

Compared to RSVP [46] which has used hop-by-hop refreshes, the default ratio of the refresh message rate to MRS refresh rate is three; we could therefore expect that end-to-end refreshes (without a feedback mechanism) for message routing states should have a higher ratio between the two rates. Hence the hop-by-hop state refresh approach will be used for the message routing state when no fast path-change detection mechanisms are available. When fast path-change detection mechanisms are applicable, they would be used instead; and the MRS would be sharing the fate of the SBS (i.e. if the SBS expires the MRS will expire as well).

3.5.1.2 Signaled behavior state refresh mechanisms

Signaled Behavior States (SBS) (i.e. state related to NAT binds or firewall pinholes) should have refresh timers comparable to existing behavior state on stateful Middleboxes (which is the case of NATs and stateful firewalls today).

For example most stateful firewalls remove a UDP stateful pinhole after 5 minutes of flow inactivity. We can therefore infer that SBS refresh periods should be in minutes and hence have low overhead (as opposed to message routing state refreshes when no fast path-change detection mechanisms are available).

In addition since SBS refreshes could be used by the PDSP SI to update the behavior it is more suitable to use an end-to-end refresh method.

3.5.2 Functional split between the CPDSL and the PDSAPs

The reason for splitting the path-directed protocol suite into two layers is to simplify the path-directed signaling applications. Path-directed signaling applications will only need to handle the signaling of specific functions in the network without taking care of how messages are sent and traverse the appropriate PDSP Signaling Forwarder(s).

The CPDSL layer provides a transparent messaging service to the PDSAPs. The PDSAPs make requests on the CPDSL layer to transfer messages with specific characteristics (Source and destination IP address, source and destination port, DSCP and other IP/transport protocol parameters) which constitute the Message Routing Information discussed in Chapter 4) and message transfer handling (including reliable and secure delivery) which constitute the Message Transfer Requirements also discussed in Chapter 4. Because the CPDSL's main function is the message transfer to the next appropriate PDSP signaling forwarder, the CPDSL layer's messaging transfer function only interacts with its direct CPDSL neighbor nodes (i.e. the scope of its interaction is with its upstream and downstream nodes and not beyond).

In section 3.5.1, we have discussed the refresh options for the two types of potential stale state on PDSP nodes relevant to PDSP protocols. We have demonstrated that the signaled behavior state should be refreshed end-to-end by the PDSAP protocol as the refresh message could be used to update the signaled behavior.
However we have not discussed in which layer the message routing state refresh mechanism should be integrated.
Since message routing state refresh is related to PDSP peer discovery and removal/refresh of the PDSP node adjacencies (for a specific data flow), the message routing state should reside within the CPDSL.
However which refresh method would be suitable for refreshing message routing state?

- Should a specific PDSAP be used as part of the CPDSL for end-to-end refresh with a feedback mechanism as discussed in section 3.5.1, or
- should a hop-by-hop refresh scheme be used instead?

In section 3.4, several user application level path-change detection mechanisms were discussed. As most applications requiring Firewall and NAT traversal support these path-change detection mechanisms, message routing state does not need to be very frequently refreshed, hence the refresh mechanism for the message routing state (either end-to-end or hop-by-hop) would not be often used. Therefore we shall avoid adding a sub-component in the CPDSL for the single purpose of end-to-end message routing state refresh. Furthermore since fast path-change detec-

tion mechanisms are available for most user level applications, we believe that fate sharing the MRS with the SBS would be sufficient, hence no MRS refresh capabilities would be required in the CPDSL layer.

3.6 Path-Directed NAT and Firewall Signaling security threats

We can classify the threats to Path-Directed NAT and Firewall signaling into two categories:

- On-path attacks: the attacker is on the signaling message path. On-path attackers could possibly drop or change messages when performing active attacks on the signaling messages. Since signaling messages should normally be changed by Middleboxes performing NAT it is difficult to determine if signaling messages were changed by an attacker or by trusted Middleboxes performing genuine NAT functions. Transitive trust with the signaling peers would be required as we shall see in the next chapter to address message integrity attacks. On-path attackers could analyze signaling messages to attack the network (for example by opening pinholes or creating NAT binds). The latter type of attack could also be performed by analyzing data flow traffic traversing firewalls and could not be mitigated by the NAT and Firewall PDSP (neither at the CPDSL nor the PDNFS layer) unless performing data source authentication at the entry of the protected network infrastructure. In addition on-path attackers could try to take ownership of the signaled behavior state as well as creating DoS attacks with signaling message floods. The topology insensitive nature of PDNFS opens the protocol to several threats; the PDNFS SI does not know in advance the Middleboxes that exist in the network infrastructure. Consequently the NATs or firewalls (i.e. the PDNFS SFs) discovered by PDNFS might not be trustworthy Middleboxes and could be attackers trying to use PDNFS to attack the network infrastructure.

- Off-path attacks: the attacker is not on the signaling message path. An off-path attacker could try to take control of existing signaled behavior states only if they were able to analyze data flow packets. These attackers could also perform signaling message floods during or before the creation of signaled behavior state.

Since the framework might[5] rely on transitive trust between the peers, if a trusted peer has been subverted the security of the NAT and Firewall PDSP will be compromised and could only be handled appropriately if proper detection mechanisms are available (for example using the Trusted Computing framework [65], to verify the integrity of the software executed on the Middleboxes, or other detection mechanisms).

We shall assess all the above threats while describing our proposed CPDSL and PDNFS PDSAP.

3.7 Summary

In this chapter we have analyzed the requirements for a global Internet deployment of a NAT and Firewall signaling protocol. The adopted protocol should be sufficiently flexible to be used across various network deployment patterns and during all stages of incremental deployment.
Our categorization of signaling protocols into two families, the targeted signaling and the path-directed signaling protocols, and their comparison has demonstrated that the NAT and Firewall signaling protocol should be a path-directed signaling protocol.
Due to the various common functions used by path-directed signaling protocols, we concluded that a two layer path-directed signaling protocol framework is appropriate. The two layer signaling protocol framework includes the Common Path-Directed Signaling Protocol (CPDSL) layer and the Path-Directed Signaling Application Protocols (PDSAPs). A specific PDSAP will need to be defined for NAT and Firewall signaling, the PDNFS PDSAP.
Challenges and issues to resolve for such protocols have been identified and will be addressed in the subsequent chapters.

[5]In case the direct trust model is not used, the transitive trust via trusted third parties would be used

Chapter 4

The Common Path-Directed Signaling Layer (CPDSL)

In this chapter we shall define the Common Path-Directed Signaling Layer introduced in Chapter 3. This layer will comply with the requirements analyzed in sections 3.3 and 3.5.
Part of the work described in this chapter is heavily inspired by the NSIS framework [16] and GIST protocol [17] developed in the IETF (for which the author made several contributions and had long private discussions with the GIST specification's authors). The author has gone beyond the NSIS WG scope to analyze all the options for the CPDSL to reduce the complexity of PDSAPs and the overall PDSP framework.

As discussed in the previous chapter, we are proposing to split the PDSP protocols into two layers, the Common Path-Directed Signaling Layer and the Path-Directed Signaling Application Protocol layer. We have seen that the CPDSL layer could have two variants:

- Variant with two independent sub-layers that would co-exist, the message transfer sub-layer used by the PDSAP and the end-to-end message routing state refresh sub-layer
- Variant with a single component which handles message transfer

In section 3.5.2, we provided our reasoning for preferring the second CPDSL variant in this work; this variant that we will call the Path-Directed Message Transfer Protocol (**PDMTP**), will be the focus of this chapter.

4.1 Terminology

New terminology introduced in this chapter:

- Message Routing Information (**MRI**): information used to determine how a message should be sent by the PDMTP, the information could include a combination of (and is not limited to) the message destination's address (or address and port), the message source's address (or address and port) and other IP packet fields used by routers' classifiers such as Differentiated Services (DiffServ) Code Point (DSCP) [66] or the IPv6 flow label [67].

- Messaging Association (**MA**): a construct created to provide communications with a specific set of characteristics - for example, some or all of reliable message delivery, congestion control, message integrity protection and confidentiality - requested in the Message Transfer Requirements for all signaling messages exchanged between two PDMTP neighbors that can be used by the signaling messages related to any data flow which requires this set of characteristics. Depending on the characteristics required, Messaging Associations could be implemented in various ways, including but not limited to: a TCP connection, a TCP connection over an IPsec [68] Security Association, a TCP connection protected with TLS [69]

- Message Transfer Requirements (**MTR**): The characteristics of the message exchange needed by the signaling for a data flow, such as some or all of reliable message delivery, congestion control, message integrity protection and confidentiality. These requirements are independent of the characteristics of the data flow to which the signaling applies.

- Path-Directed Message Transfer Protocol (**PDMTP**) : the single constituent of the CPDSL providing PDSAP peer discovery and message transfer.

- PDMTP neighbor: next PDMTP hop node (upstream or downstream) on the path between SI and SR. This node may be several IP hops away.

- Message Routing State (**MRS**): ephemeral information (i.e., it is implemented as soft state), specific to an MRI, used to keep track of the associated PDMTP neighbor to which

the message should be routed and for which a Messaging Association should potentially be used while sending messages for the specific MRI

- Message Routing State Unique Identifier (**MRSUID**): a probabilistically unique randomly generated identifier for a MRS associated with a specific user application data flow
- PDU : Protocol Data Unit
- PSI : PDMTP Signaling Initiator
- PSR : PDMTP Signaling Responder
- MA type: identifies the protocol(s) used for the MA for example TCP, TCP/TLS, IPSEC/IKE/TCP

4.2 PDSAP Interactions with the PDMTP

The PDMTP internal operations are completely hidden from the PDSAP (PDNFS or other PDSAPs).

Every signaling application (i.e. PDSAP) provides the PDMTP with the Message Routing Information (as defined in section 4.1) needed to deliver the messages to their intended destination and the MTR (privacy protection, integrity protection, ordered and reliable delivery, etc) that specifies the preferred characteristics of the exchange when it sends a message for delivery by the PDMTP. Every PDSAP message could be handled totally independently by the PDMTP but in many cases it will be more efficient for the PDMTP to handle all the messages relating to a particular data flow for a specific recipient using common resources.

4.3 PDMTP Overview

For a given PDSAP, the path between the SI and the SR will contain a number of Middleboxes that provide functionality relevant to the PDSAP and some that do not. For example a simple path between two end-hosts through a service provider network might traverse two NATs,

four firewalls and a QoS gateway. Assuming we have one PDSAP that deals with NATs and firewalls (such as the one specified in Chapter 5) and a second PDSAP that deals with QoS, we have seen that it is essential that the NAT and firewall signaling interacts with the various NAT and firewall Middleboxes sequentially as the signaling message travels along the path, but these messages have no relevance to the QoS gateway. Hence we can segment the path, as it concerns a particular PDSAP, into a sequence of point to point links between the Middleboxes which implement the PDSAP and provide functionality associated with it. The PDMTP should therefore operate in terms of these path segments. Starting from the SI, the PDMTP should provide the ability to discover the next SF on the path that implements the relevant PDSAP and provide the capability to transfer the signaling messages across this segment of the path delivering it to the PDSAP layer on the SF. This functionality can then be used sequentially to build up a sequence of path segments that eventually lead to the SR. The PDMTP operates essentially independently across each segment with requirements being passed through the PDSAP layer at each Middlebox that implements the PDSAP with Middleboxes that do not implement that PDSAP being bypassed once the PDMTP has discovered the next waypoint or neighbor on the path. In the example given, the PDMTP would establish a sequence of seven segments between the SI and the SR visiting the six NAT and firewall PDSAP instances for NAT and firewall signaling and (if required) would establish a separate set of two segments covering the path and visiting the single QoS gateway. Hence we can specify the PDMTP as a protocol that has to discover the correct downstream neighbor that implements the relevant PDSAP using the MRI provided by the PDSAP and then establishes a point to point link to deliver signaling messages between these neighbors according to the MTR provided by the PDSAP. The PDMTP handles the discovery of PDMTP neighbor nodes and transfers PDSAP messages according to the Message Transfer Requirements provided by the PDSAP layer.

There are two modes of operations for a PDMTP node:

- An initiator mode, where messages are sent to the correct downstream node which is next along the path related to a specific data flow having initially discovered the identity of this node by sending PDMTP peer discovery messages. In this mode the PDMTP node operates as a PDMTP Signaling Initiator (PSI) and initiates the discovery phase

providing the next downstream adjacent PDMTP node for the specific flow.

- A responder mode, where messages are sent to upstream nodes identified following the reception of PDMTP peer discovery messages. In this mode the PDMTP node operates as a PDMTP Signaling Responder (PSR).

Depending on the node type it might use one or both modes of operation while operating on the same flow direction (either downstream towards the data receiver or upstream towards the data sender). An application aware host would normally support the initiator mode for the downstream direction and the responder mode for the upstream direction (i.e. for the received data flow). Middleboxes would normally support both the initiator mode (for the downstream direction) and the responder mode (for the upstream direction) with respect to the associated data flow. We shall describe first the initiator mode, where upon reception of the signaling application PDU the PDMTP layer follows the following process:

- Step 1: the PDMTP layer determines if it already knows to which downstream PDMTP neighbor it should send signaling messages of a specific PDSAP[1] with the provided MRI and the specific MTR. A lookup is performed in the Message Routing State table which is maintained for that purpose. If an entry is found and it matches the required MTR then the process continues with step 4. The PDSAP request may instruct the PDMTP layer to perform the discovery of the PDMTP downstream neighbor even if there was an existing entry in the MRS table, in which case, or if no matching entry was found, the process continues with step 2.

- Step 2: If no entry is found or the entry is expected to be no longer valid or the PDSAP requires the downstream PDMTP neighbor discovery, a PDMTP downstream neighbor discovery message (we shall call it PDISC for Peer Discovery) is sent to the MRI's destination and to a specific port number (discussed in subsection 4.3.1) encapsulated in a UDP datagram (to minimize the amount of required state) and with a specific IP option (the IP option value is further discussed in section 4.3.2). The PDISC message includes

[1]Since PDSAPs may not always be collocated, the node could have a different PDMTP neighbor for a different PDSAP even if it was for the same MRI and MTR

the address[2] to reach the PSI instance from which it is being sent and a cookie, the PDMTP-I cookie. The cookie needs to be echoed back in response messages to correlate the PDISC message with its response. The PDISC includes an additional parameter used to correlate received messages with the associated MRS entry, this parameter is the MRSUID (MRSUID) with a large value space. The MRSUID will be used in all subsequent PDMTP messages relevant to the same MRS.

The PDISC message includes one or more protocol proposals (we shall call this list of protocols a stack proposal) that could be used to meet the MTR (if more than simple datagram transport is required) expected by the PDSAP application. The downstream PDMTP neighbor that has registered an interest in the IP option used (in section 4.3.2 we shall discuss how a PDMTP downstream neighbor decides to intercept PDISC messages based on IP option values) in the IP packet embedding the PDISC message will intercept the message and analyze the MRI. The intercepting node will do a lookup in its Message Routing State table against the received MRI with the required MTR. If an entry is found and an existing valid Messaging Association is already established with the upstream node sending the PDISC, the intercepting downstream node will respond back (with a message that we will call PDISC-RESP) through the existing Messaging Association. If no existing entry is found in the MRS table then the intercepting node (the PSR) responds back with a PDISC-RESP message encapsulated in a UDP datagram and addressed to the upstream PDMTP node. The later PDISC-RESP message provides the supported Messaging Association protocols and the original stack proposed by the upstream PDMTP node. If the PDISC message from the upstream node does not contain a stack proposal, the intention is that messages should be exchanged using UDP encapsulation only. If the responding node's local policies do not allow this, the downstream node returns an error message; otherwise the PDISC-RESP message also does not contain a stack proposal. To avoid state poisoning attacks on the downstream nodes, the intercepting downstream node (the PSR) does not immediately create a new entry (or update an existing entry) in its MRS table; instead the intercepting node will

[2]The PSI instance address is required since the message's source address is not necessarily (this will be discussed in section 4.3.1) the same as the PSI address

send a cookie, the PDMTP-R cookie which will be used for reverse routability checking. In addition to the PDMTP-R cookie, the PDISC-RESP includes the PDMTP-I cookie provided by the PDISC message.

- Step 3: the upstream PDMTP node (the PSI) that previously sent the PDISC message for the specific MRI, is expecting to receive a PDMTP-RESP message (if no message was received after a certain time, the PDMTP initiator retries step 2 at most MAXRETRYPDISC times (we suggest retransmitting twice at most). The later message could be either sent over an existing Messaging Association established with a previously discovered downstream peer, or be sent encapsulated in a UDP datagram since no Messaging Association was previously established (or met the requirements implied by the MTR).

  The PDMTP node responds back to the downstream node's PDISC-RESP message with a MRS-EST message (sent to the address of the discovered downstream node) confirming that the message routing state was installed on the node and demonstrating that the reverse routability[3] check required by the downstream PDMTP node was successful (by copying the PDISC-R cookie). If the PDSAP's MTR required a Messaging Association and none was available with the discovered node then a new Messaging Association is established based on the mutual stack agreements implied from both peers' stack proposals; the MRS-EST is then sent on the newly established MA. If an existing MA was available and matches the PDSAP's MTR then the MRS-EST is sent over it. Upon reception of the MRS-EST message, the discovered downstream node verifies that the message is associated with an existing MRI within the MRS table as well as the corresponding PDMTP-R cookie previously sent in the PDISC-RESP message. After positive validation (which includes, when applicable, the matching of the PSI and PSR node identities with the credentials used to establish secured MAs) the PDMTP node installs the MRI's associated MRS table entry, and from that moment onwards messages could be sent by both PDMTP peers. On the upstream and downstream PDMTP peers, the MRS entries associated with the specific MRI would have MA entries (when applicable), we shall discuss

[3] a standard mitigation technique employed against DoS attacks using address spoofing

the interaction between the MA entries and the MRS table in section 4.5.

- Step 4: the PDMTP node transmits messages based on the MRS table entry.
- Step 5: To detect (and trigger the path-refresh) the divergence between the data flow path and the PDSAP signaling message path, the end-host's path-change detection mechanisms are used (described in section 3.4 and relies on the user application's data flow packets). The MRS entry shares the fate of the PDSAP's signaled behavior state (as discussed in Chapter 3). Hence the MRS entry will be removed when the PDSAP's signaled behavior state expires or is explicitly removed. The PDSAP layer might request the refresh of the MRS entry in case the PDSAP layer was notified of a path-change.

The MRS refresh or removal trigger will be provided to the PDMTP by the PDSAP (via the PDMTP/PDSAP API) which understands the properties (data flow path-change detection mechanisms) of the user application. Figure 4.1 summarizes the PDMTP initiator mode of operation.

The PDMTP Responder mode of operation, shown in Figure 4.2, is implied by the PDMTP Initiator mode of operation; the roles are inverted:

- The node is expecting to receive PDISC messages
- The node will respond by sending a PDISC-RESP message over either an existing MA or a new one (if an MA is required)
- The MRS entry is created only at reception of a valid MRS-EST message (with a valid PDMTP-R cookie received) over a MA (when applicable)

Figure 4.3 shows a simplified message sequence of the MRS entry's initial creation stages prior to the Messaging Association establishment. Figure 4.4 shows the message exchanges during and after the Messaging Association establishment.

4.3.1 Message Routing Information Properties

When deciding to send a PDSAP message, the PDSAP layer needs to communicate to the PDMTP layer the necessary MRI and MTR information. There are two main requirements for the PDSAP layer when determining the MRI:

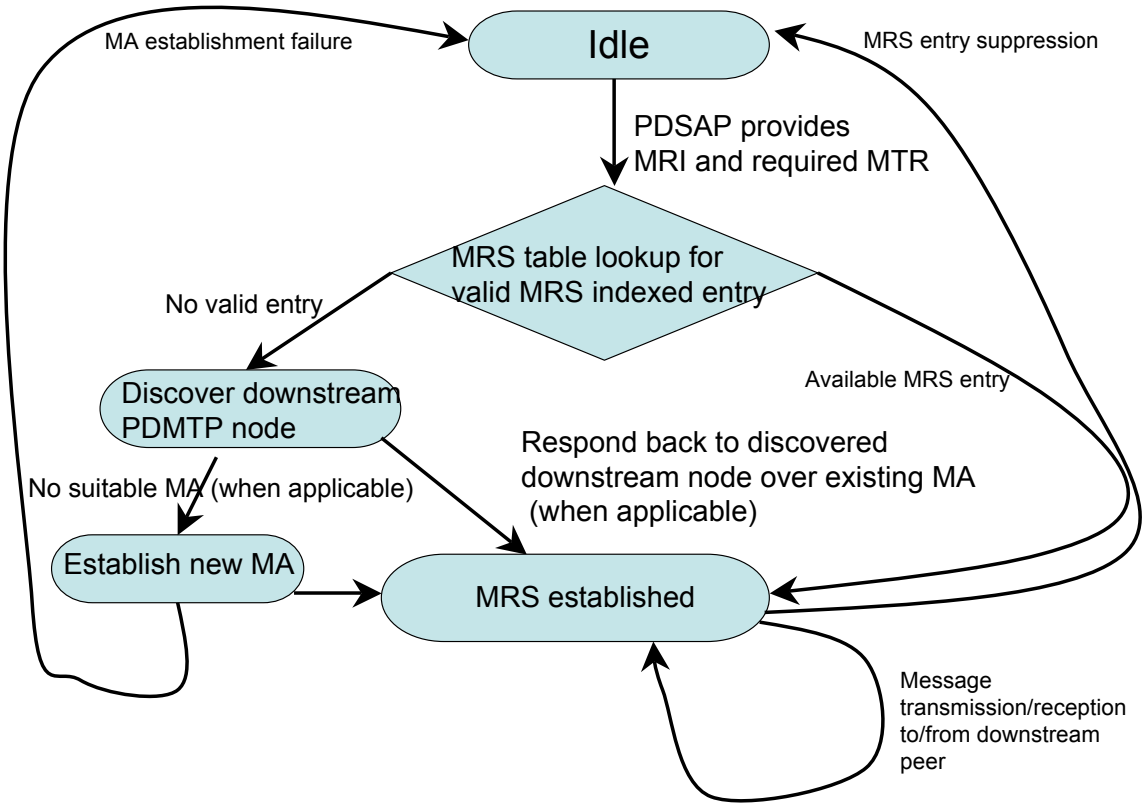


Figure 4.1: PDMTP Initiator mode of operation

- The chosen mode of operation for the PDSAP protocol (introduced in section 3.3.1): path-coupled or path-decoupled. When the path-coupled mode is used, depending on the use of load balancing schemes in the network, the PDSAP protocol potentially might not traverse the same Middleboxes as the data flow packets. As we have seen in section 3.3.3, most ECMP implementations on routers use only the source and destination address for load balancing; hence for the signaling messages to trace the data flow's path it would be sufficient to use the same source and destination IP address as the data flow.

  The transport port used for either the source or message destination would not impact the chosen path in almost all cases of which we are aware. Accordingly we propose to use arbitrarily chosen source transport ports and a well known message destination transport port (to avoid conflicts with ports used on PDMTP unaware end-host systems using the port for different protocols).

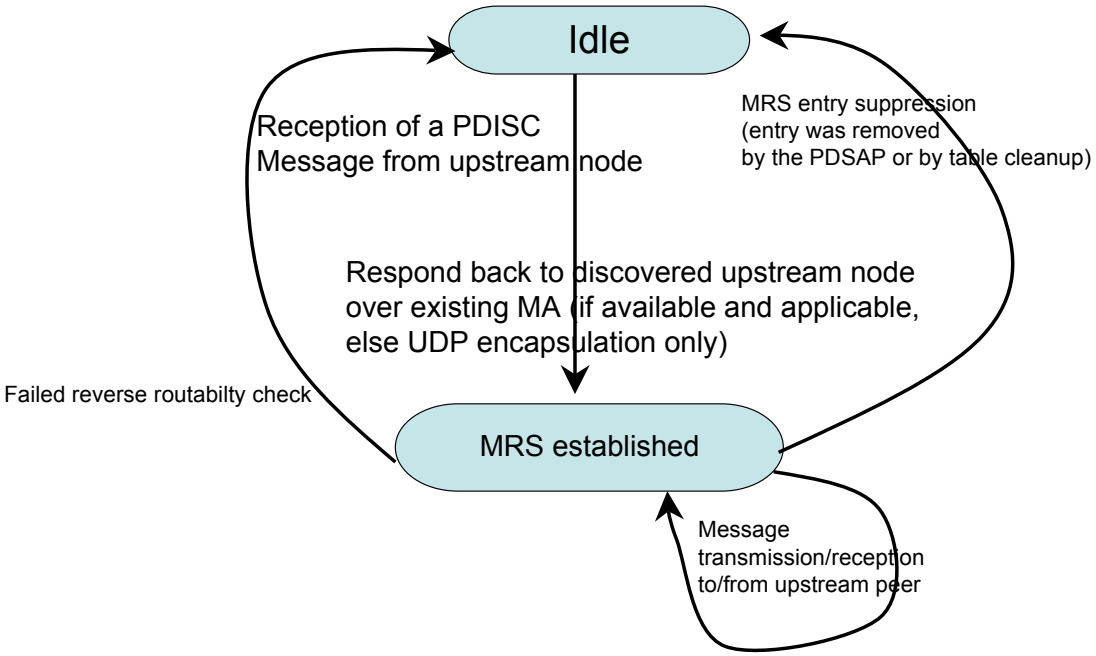


Figure 4.2: PDMTP Responder mode of operation

If the path-decoupled mode of operation is required, the MRI would be set independently of the data flow source and destination address and transport ports; this type of message routing should only be used when applicable (small and simple networks or for the PDTS PDSAP discussed in Chapter 6). In some cases as discussed in Chapter 6, the destination address could be the data flow address (or an address expected to allow the message to go towards the data flow recipient's address).

- Impacts of use of data sender address as source address by proxies and PDSAP nodes: In order to ensure that signaling messages follow the same path as the data flow to which the signaling applies, proxy SIs and other PDSAP nodes need to send PDISC messages with the data sender address as the legitimately spoofed source address. This might result in any ICMP error messages resulting from the PDISC message not being received by the node which sent the signaling message, as they would be sent to the spoofed data sender

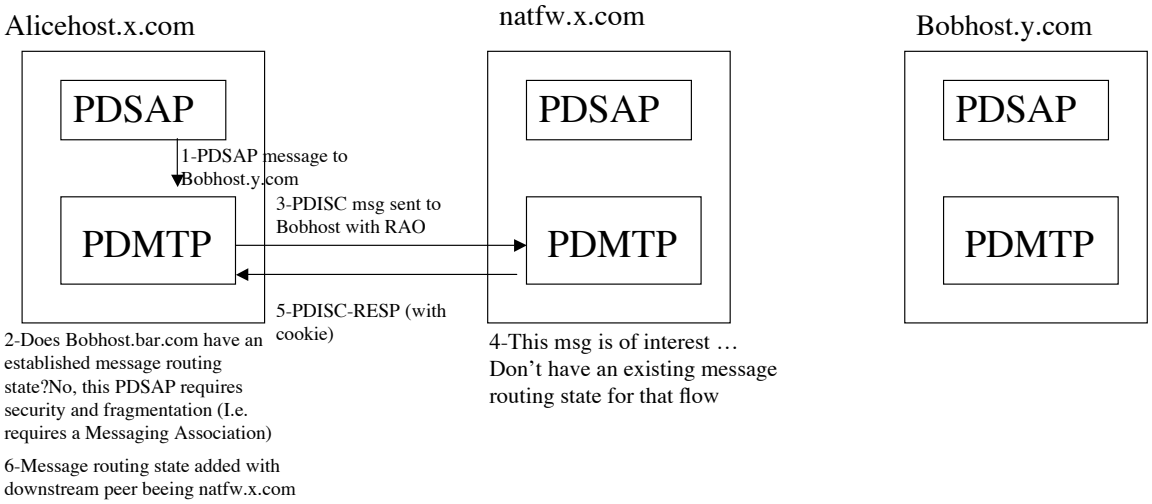


Figure 4.3: PDMTP message sequences prior to MA creation

address without a RAO. This would mean that the ICMP messages were not intercepted [4] or processed above the IP layer even if the message passes through the sending PDSAP node.

Reception of these errors could be very important to avoid delays impacting real-time applications such as VoIP. It is also possible that the PDISC message might encounter anti-spoofing mitigation techniques in the Internet (for example ingress filtering discussed in [70, 71]. There is a small possibility that the spoofed source address could lead to the PDMTP messages being dropped.

However we do not foresee that this would be a significant problem because in almost all cases the signaling and data flow packets would arrive on the same interface, so satisfying the ingress filtering rules.

[4] because of possible asymmetric routing

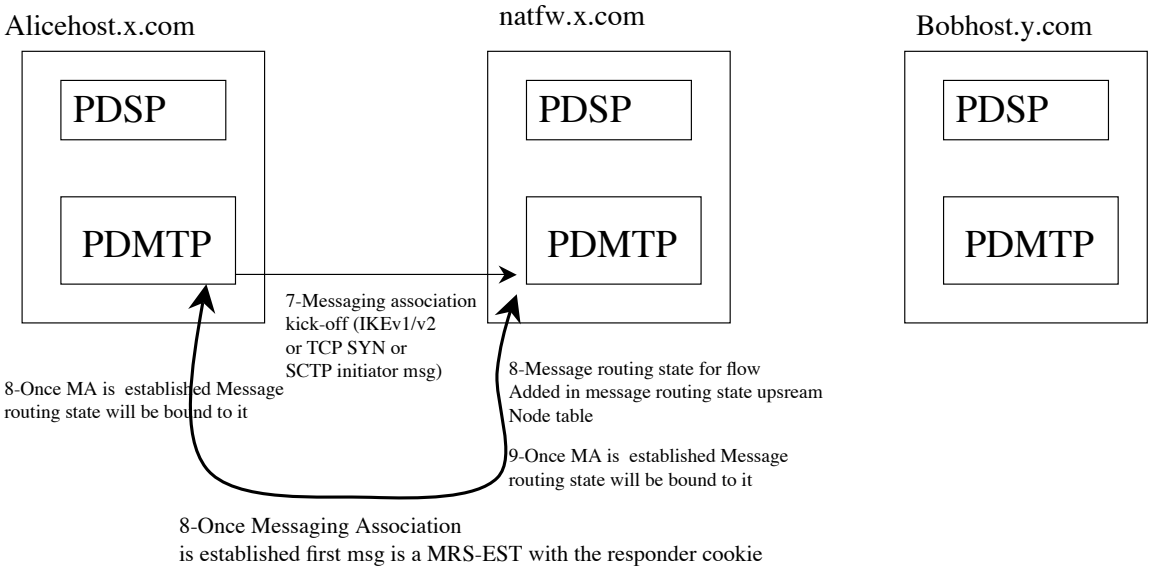


Figure 4.4: PDMTP message sequences

4.3.2 Router Alert Option Setting

The Router Alert Option [47, 48] is the interception mechanism used by the PDMTP protocol. Since the PDMTP protocol is designed to be the common message transfer protocol for several PDSAPs, an efficient and fast method should be used to indicate to the PDMTP layer if it should intercept messages that are not addressed to the PDMTP node based on the router alert value.

Before determining how to set the appropriate RAO values for each PDSAP, we shall first analyze the circumstances in which a PDMTP node might need to intercept a message:

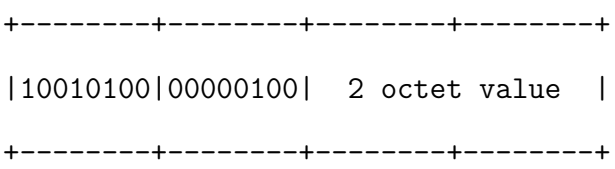
- The PDSAP protocol uses addressing referrals (we shall discuss this further in section 4.6). Even if the PDSAP protocol is not supported on the Middlebox traversed, if the Middlebox applies NAT or NAPT processing on packets traversing the node, all PDSAP protocols will need to be intercepted because they carry the specification of a data flow

that will be traversing the node (a 'referral').

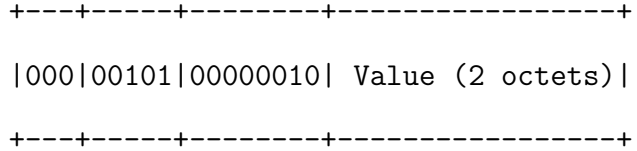
- The PDSAP protocol is supported on the node traversed. There could be specific cases where a PDSAP protocol is supported by the node traversed but not the signaled behavior associated with specific PDSAP messages; hence the granularity of the interception should make it possible to distinguish not only a specific PDSAP protocol but also specific PDSAP messages within that protocol.

Based on the above interception requirements we may conclude that the RAO should be selected based on recommendations of the PDSAP layer since the PDMTP layer does not understand the content of the PDSAP messages. Hence the setting of the RAO values would need to be indicated via the PDSAP$<->$PDMTP API.

Although the IPv4 RAO and IPv6 RAO are different (shown below), they share the same two octet RAO value space (shown in the figure below), for which only a very few values have been previously assigned [72]. Due to the large RAO value space we propose to allocate RAO options both for PDSAPs and specific PDSAP messages (the PDNFS protocol, discussed in Chapter 5 will benefit from this feature. A list of RAOs of interest and associated PDSAPs (supported ones and the ones having NAT issues in case the node had a NAT daemon) must be used by the PDMTP, this list could be incarnated in a table on the node.



RAO in IPv4



RAO in IPv6

4.3.3 PDMTP Message Encapsulation Summary

A PDMTP message could either be simply encapsulated as the payload of a UDP datagram or may need to be carried in the payload of more complex protocols that are required to match the requirements of the MTR provided by the PDSAP layer. The PDMTP peer discovery procedure can only use a UDP datagram encapsulation with a specific RAO value, hinted or directly assigned by the PDSAP layer via the PDMTP$<->$PDSAP API because this is the only transport protocol that can be used prior to validating the identity of the next PSR and establishing state on that node.

Table 4.1 provides a summary of the encapsulation schemes used for PDMTP messages.

Table 4.1: PDMTP message encapsulation Summary

Message Types	**PDISC**	**PDISC-RESP**	**Other messages**
Encapsulation method	UDP and RAO	MA[5] if already exists else UDP	MA[5]

PDMTP messages used to establish or refresh MRS states could also carry PDSAP PDUs depending on the PDU's related MTR. We do not foresee any use of this type of piggy-backing for the PDNFS PDSAP (except for the MRS-EST messages) but other PDSAPs could potentially use this capability.

4.3.4 MRS table cleanup

In Chapter 3, we have proposed not to include MRS refresh capabilities in the CPDSL layer (and therefore in the PDMTP protocol).

Due to the layer split architecture, the PDMTP implementation should protect itself from the failure of the PDSAPs with which several MRS might be fate sharing with associated SBS. Therefore a MRS table cleanup mechanism should be used.

We have avoided the use of MRS refresh timers to avoid devoting CPU resources to that purpose, hence the MRS table cleanup should consider the current PDMTP activity before flushing the MRS table entries. Otherwise significant processing resources will need to be used

[5]Only if the MTR required the use of an MA

to create simultaneously (depending on the PDMTP traffic periodicity) the new MRS entries. The table cleanup activity is not associated with any particular MRS entry, hence the impact on processing load can be made relatively negligible depending on the selected cleanup period and the PDMTP traffic activity threshold used by treating it as a background activity. For example, it would not be necessary to cleanup the whole table at once. Instead portions of the table could be checked in turn whenever processing resources were available subject to memory resources not being exhausted if the cleanup period was overly long. This measure will avoid using significant processing resources in case simultaneous MRS entry creations would be required.

The cleanup timer (for each part of the table) should be set based on an estimated average overall lifetime of a SBS and the average PDSAP inter message sending/reception. A simplistic approach could consider the node's local time to select the cleanup time in predictably quiet PDMTP traffic periods (late in the night for example).

Alternatively the flushing of idle entries could be based on appending to the MRS entry a time stamp indicating the last time the entry was used. Upon performing the table flushing, entries having a time stamp value less than the current time minus a MRSIDLE-TIME value would be flushed out. We believe that this time stamp based solution is more appropriate for cleaning stale entries from the MRS table.

4.4 PDMTP Signaling Initiator and PDMTP Signaling Responder Interactions Within the Same PDMTP Node

To simplify the PDMTP protocol implementation we believe that it is important to avoid any direct interaction between PDMTP Signaling Initiator and PDMTP Signaling Responder instances within the same node.

However the PDMTP Signaling Initiator instance on the node would benefit from certain information that the PDMTP Signaling Responder might be aware of.

This raises one question: how would the PSI and PSR instances (indirectly) exchange information? We believe that the following proposal is applicable:

- Use a single MRS table for the PDMTP node, as shown in Figure 4.5. This table would have entries managed by both the PSI and PSR instances. With the PDMTP SI instance owning the entries relevant to the downstream node adjacency and the PDMTP SR instance owning the entries relevant to the upstream node adjacency. The use of this single MRS table will accelerate the explicit removal of MRS entries when requested by the PDSAP layer, the PDSAP layer will simply request the removal of PSI and PSR instances associated with a specific MRSUID.

MRSUID	PDSAP	Source	Destination	Upstr	Upstr MA	Downstr	Downst MA	Timestamp (UTC)
MRSUID1	PDNFS	10.1.2.3	172.16.0.4	A	MA1	B	MA2	977616000
MRSUID2	PDQS	10.1.2.3	172.16.0.4	A	MA1	C	MA3	977615980

MRS table on natfw.foo.com

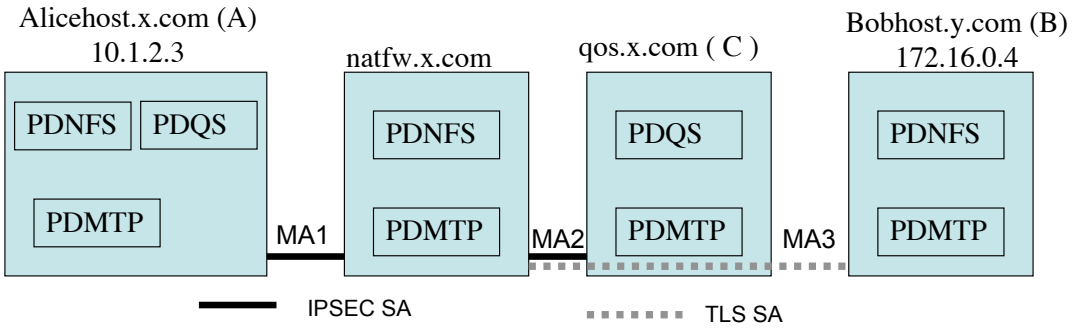


Figure 4.5: A single MRS table for the PSI and PSR instances on the same PDMTP node

In Figure 4.5, we show a possible simple representation of the MRS table, where two PDSAPs are used: the PDNFS and the PDQS for signaling quality of service. As discussed in section 4.3.2, a PDMTP responder should intercept messages not only because the PDSP node supports the signaled PDSAP but also for other reasons such as embedded addressing information (which would be the case for a PDQS intending to provide proper QoS for specific flows).

We shall discuss further the reasoning for the latter interception in section 4.6. In the next section we discuss the Messaging Association entries (MA1, MA2, MA3 in Figure 4.5) in the MRS table and consider the life cycle of Messaging Associations on a PDMTP node.

4.5 Messaging Association Usage and Life Cycle

A Messaging Association is established by the PSI when a PDSAP requests a specific MTR for a message with a specific MRI for which UDP datagram encapsulation will not suffice and for which there are no established Messaging Associations with the downstream peer which is discovered for this MRI during the discovery phase. Messaging Associations are designed to provide a bidirectional connection channel between an (upstream) PSI instance and a (downstream) PSR instance. In principle, an established Messaging Association can be used in the reverse direction to that for which it was originally established, i.e., by a PSI instance on the node which is hosting the PSR originally involved in setting up the connection.

The mapping of a MTR set to an appropriate set of transport protocols and security capabilities is left to the PDMTP layer which is aware of the transport layer protocols and security frameworks (i.e. network and transport layer security) supported on the node. In most cases an MA will be based on a reliable connection oriented transport protocol (such as TCP) since messages that do not need reliability would not necessarily need the overhead of a Messaging Association. We shall describe below how the type of Messaging Association to be used is negotiated between PSI and PSR and how MAs are established and bound to MRS entries.

When the type of Messaging Association required is determined by the PSI instance, it can assemble and send the PDISC message indicating to the prospective peer one or more sets of protocols and security capabilities, known as '(protocol) stack proposals' which the PSI could use for the Messaging Association. Each possible stack proposal supported by a node corresponds to a 'MA type' which can be used to classify the Messaging Associations in the MA table in PSI and PSR instances. Operations can then proceed as presented in section 4.3 to establish any necessary Messaging Association and exchange signaling messages. Once a PSR instance receives a PDISC message with a set of stack proposals, it verifies that it supports at least one of the proposed MA types. If it does not support any of the proposed MA types it returns

an error indication to the originating PSI which must notify the requesting PDSAP that the message could not be sent with the requested MTR. Assuming the PSR instance supports at least one of the proposed MA types, it queries its Messaging Association table to determine if it contains an entry for the PDMTP upstream peer that sent the PDISC message that matches any of the MA types proposed in the new discovery message.

If there is no match for the query, the PSR responds back with a PDISC-RESP message containing both a copy of the list of MA types proposed by the PSI and a PSR preferred MA type selected (based on the node's local policy in case a choice had to be made) from the mutually acceptable MA types. The information is encapsulated in a UDP datagram addressed directly to the originator of the PDISC message as discussed in section 4.3

If a match is found for the query, the PSR responds back with a PDISC-RESP sent over the MA identified by the query as discussed in section 4.3

When a PSI receives a PDISC-RESP over an existing MA, the PSI binds the existing MA to the MRS entry in the MRS table. Alternatively, when a PSI receives a PDISC-RESP message other than over a Messaging Association (i.e., the message is encapsulated in a UDP datagram), it uses the MA type selected by the PSR (returned in the PDISC-RESP message) and establishes an MA between the PSI and the PSR.
Once the MA is established it is added to the MA table with the MA type (used transport protocols and security proposal: IPSEC or TLS). The MA is then bound to the associated MRS entry, for which it was created, in the MRS table.

The PSI instance then sends an MRS-EST message to the PSR over the pre-existing or newly established MA. The PSR instance, upon reception of a valid (based on criteria discussed in section 4.3) MRS-EST over a newly-established MA (i.e., established prior to the reception of the message but after the transmission of the PDISC-RESP), the PSR binds the MRS entry to the newly-established MA. This binding will result in a new upstream MA entry in the MRS table, as well as a new MA in the MA table.

We show in Table 4.2 a representation of a MA table on the natfw.x.com node shown previously in Figure 4.5. Messaging Associations are generally expected to remain in existence for at least as long as there are active MRS entries which are using them. Messaging Associations

Table 4.2: MA table used by both the PSI and PSR instances on the same PDMTP node

MA index	**MA type**	**MA remote endpoint ID**
MA1	IPSEC/TCP	A
MA2	IPSEC/TCP	C
MA3	TCP/TLS	B

consume resources on PDMTP nodes even if there are no PDSAP messages using the MA and hence it is necessary to manage the MAs to contain the resource usage. The lifetime of MAs is discussed further in the next section.

4.5.1 Messaging Association Maintenance and Lifetime

To avoid any stale MAs (upstream or downstream) on a PDMTP node, established MAs need to be periodically validated. This may be possible using the constituent protocols of the MA (for example TCP keep-alives). This might not be always possible or too slow without operating system kernel modifications[6], therefore the PDMTP protocol needs to provide keep-alive and validity checking capabilities. We propose to use a simple HELLO message sent by the PSI which originally created the MA to the corresponding PSR. Assuming that the MA is using a reliable, connection oriented transport protocol, such as TCP, this single message will effectively validate both directions of the connection, but we should be aware that failure of the reverse direction may not be detected until a second HELLO message is to be sent without being acknowledged.

HELLO messages should only be sent after the expiry of an idle timer, for which we recommend a default value of ten minutes. The PSR should terminate the MA if it fails to receive any kind of message on the MA for a period of three times the idle timeout (i.e., 30 minutes). Since all PDMTP messages use a MRSUID to associate the message with a specific MRS, we propose to use an "all binary ones" pattern MRSUID for HELLO messages which will not be used for ordinary MRSUIDs.

[6]On Freebsd the TCP keep-alive interval can not be configured with the sysctl command; other operating

The lifetime of an MA is dependent on its use, the PDMTP nodes caching policies and the PDSAP inputs. Establishing MAs takes time and therefore would impact real-time applications using PDSPs if they have to be established for every new PDSAP message or flow. To avoid these issues an MA should generally remain active depending on local policies even if it is not immediately in use. These local policies could consider the following parameters for the lifetimes of MAs that are not actively in use:

- History of MA usage: if the MA has been bound to several MRSs, the idle timer could be set greater than if it had only been used a single time. This option requires the PDMTP to keep track of the usage across the various MRS bound to the MA.

- Node location in the topology and the nature of the interface(s) bound to the MA. This is a set of parameters configured by the node's administrator. For example if a node is a stub network firewall (i.e. the firewall is the first PDMTP hop on the path), all interfaces on the protected network with active MAs would be bound to MAs established by SIs within the protected network. All these MAs should be maintained for a long period of time since the SIs have no other egress to the wider Internet. That period would be dependent on the credentials provided by the SI during the MA establishment (otherwise these sustained MAs could be used to launch DoS attacks). Within the same autonomous system, we envisage that PDMTP nodes would maintain their MAs for long periods of time and propose to have a default value of 24 hours before the expiry of idle MAs.

- The PDSAP could provide hints on the usage frequency of the MA and indicate to the PDMTP if caching is required.

- Resource availability:

  If a PDMTP node is running short of resources to establish new MAs, it can tear down existing MAs according to, for example, a least recently used policy. MAs can be torn down even if there are active MRSs using the MA because the PDMTP will automatically reestablish the MA when it needs to actually send a message on one of these MRSs.

systems such as AIX or Solaris allow its configuration with root privileges. In any case the keep-alive interval is generally not configurable on a per connection basis but only as a general parameter common to all connections

The discovery process will be reinvoked as necessary. This mode of operation has to be supported in any case to cater for situations where the MA becomes unusable because of routing changes.

When a MA, bound to a specific MRS, is no longer active and is removed from the MA table, upon reception of a PDSAP message via the PDMTP/PDSAP API all MRS entries associated to that MA have to be refreshed. If a new downstream node is detected, the PDMTP layer notifies the PDSAP layer which would notify the PDSAP SI and then re-trigger a complete path-refresh.

Figure 4.6 provides a summary of an MA's life cycle.

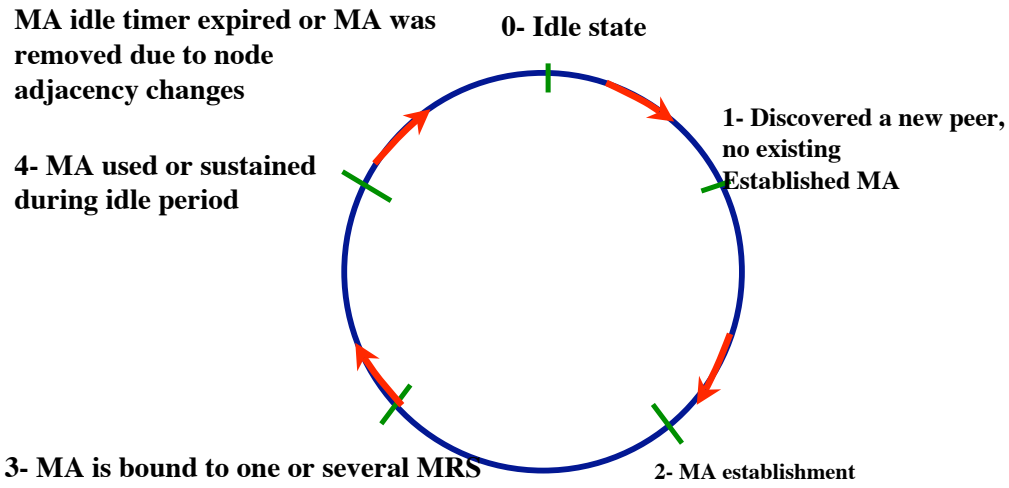


Figure 4.6: Messaging Association life cycle

4.6 NAT Traversal Implications on the PDMTP

In section 3.3.4 we have analyzed the issues encountered by PDSP protocols (i.e. the combination of the CPDSL and the PDSAPs) with NATs. We shall utilize that analysis in determining impacts of a NAT on the PDMTP layer.

In summary the expected issues are as follows:

- Message forwarding: how are PDSAP messages (encapsulated in PDMTP messages) forwarded to the appropriate PDSAP SR?
- Embedded addressing issues for data flow and signaling messages (both PDMTP and PDSAP messages), as well as for Messaging Association establishment

Several PDSAPs (including PDNFS) signal for behaviors to be applied to a specific flow (or group of flows). Every flow on which a specific behavior has to be enforced is identified by a filter matching expression (the most common one being the five tuple: protocol type, source and destination address and transport ports). When a PDSAP message traverses a NAT, if the NAT does not understand the PDSAP message, any filter matching expression embedded in the PDSAP will not be modified by the NAT (a well-known issue with NATs). Hence the behavior signaled by the PDSAP will be incorrect for any Middleboxes subsequent to the NAT as the data flow packets will be modified as they pass through the NAT and consequently the required behavior will not be enforced on the data flow packets relevant to that PDSAP message.
As the common layer to all PDSAPs, the PDMTP layer could query (directly or via the PDNFS layer) the NAT bind table and gather information on the translations that the NAT will make for the data flow associated with the message allowing it to build an updated filter expression to be used for the data flow subsequent to the NAT. Consequently all filter expressions for data flows have to be part of the PDMTP message encapsulating PDSAP messages rather than embedded in the PDSAP specific part of the message which would be opaque to the PDMTP. Although this leads us to the conclusion that the PDMTP payload should carry the filter matching expressions explicitly, we must also determine the required interactions between the PDMTP and the NAT engine (either direct or via the PDNFS if applicable). We shall analyze a number of different deployment scenarios involving situations that might arise before and after complete deployment of the signaled Middlebox control protocols to fully understand the consequences for the PDMTP layer of the impacts summarized above (message forwarding and embedded addresses issues) and so decide which approaches provide the best solution.
We consider the following possible deployment scenarios where xPDSAP represents any of the possible PDSAP functions that might be deployed:

- xPDSAP SI/SR with a downstream/upstream PDMTP aware NAT
- xPDSAP SI/SR with a downstream/upstream PDMTP unaware NAT
- xPDSAP SI/SR with a downstream/upstream PDNFS only aware NAT node.

In the following subsections we shall analyze this set of deployment scenarios to determine all the requirements on the interactions between the PDMTP, the PDNFS and the NAT engine

and then define what appears to be the optimal solution. In addition to PDMTP message forwarding and to updating the payload of messages with addresses consistent with the address translation, correct Messaging Association establishment must also be ensured. We discuss Messaging Association establishment in the subsequent sections as well as specific issues that may arise when using the Internet Key Exchange (IKE) protocol versions and the IPsec security framework.

We distinguish two types of solutions to the problem: one where a PDMTP adjacency is required at the NAT regardless of the support of the PDSAP message being transported by the PDMTP messages; and another solution type where no PDMTP adjacency is used. The latter solution type is required to solve the issues (message forwarding, embedded addresses and Messaging Association establishment), whereas the first solution solves the MA establishment problem inherently as we shall see in the following subsections.

4.6.1 PDMTP Interactions with Downstream/Upstream PDMTP Aware NAT

In this subsection we shall analyze a deployment scenario where a PDSP node has either a downstream or upstream PDMTP aware NAT node. The analysis covers both cases where the PDSP node supports the PDNFS PDSAP as well as those where it does not support it. In both cases we shall also investigate the use of the PDMTP adjacency.

4.6.1.1 PDSP Node Without a PDNFS PDSAP

We shall consider the cases where the PDSAP instance is a SI or SR deployed in a network with a private addressing scheme using a (PDMTP aware) NAT to provide global connectivity to the stub network. In the example discussed only one node (the SI or SR) is behind a Middlebox supporting NAT services, but the same concepts are applicable when both nodes (SI and SR) are behind Middleboxes enforcing NAT services. This means that in this example outgoing packets from the Middlebox (enforcing NAT services) will have globally routable addresses.

4.6.1.1.1 x PDSAP SI with a downstream PDMTP aware NAT and with the use of a PDMTP adjacency

In Figure 4.7 we show how the PDMTP layer intercepts a message sent by an "x" PDSAP but that PDSAP is not implemented on the intercepting node. Since

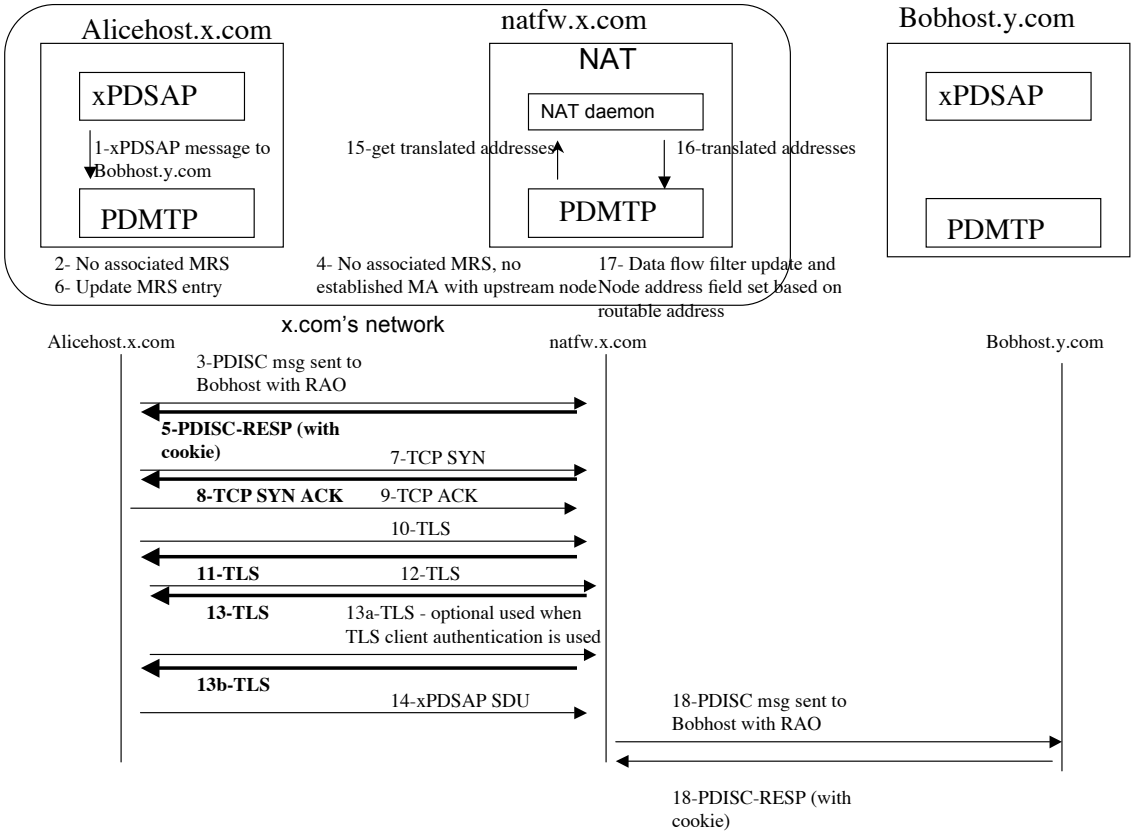


Figure 4.7: xPDSAP SI with a downstream PDMTP aware NAT

the message's destination is a routable address the Middlebox is able to route the message to the appropriate destination, hence message forwarding is not an issue in the case of this SI. The PDMTP layer has to be able to learn from the NAT engine the translated addresses of the data flow's source address as well as the signaling message translated source address. In addition the PDMTP SI (on the intercepting node in the example it is natfw.x.com) needs to be instantiated to send the x PDSAP message.

As a result the NAT engine software provides the translated addresses (source addresses and ports for the signaling message and the data flow) and by some means (determined at the end of the analysis) an instantiated PDMTP SI (PSI) will send a message downstream with

the same MRSUID communicated by the upstream PDMTP node but with the MRI using the translated message's source addresses. However the approach in section 4.4 which suggests using the same MRS table for both the PSI and PSR instances results in an inconsistency with the source address/transport port entry (created by the PSR instance on the natfw.x.com node) associated with the MRSUID. This inconsistency is due to the translation of the message's source address and (possibly) transport port by the NAT engine, however since the MRS entry could be uniquely identified by the MRSUID this problem can be overcome provided that the MRSUID is used as the correlation information to relate messages sent by the downstream (bob.y.com node in the figure) to the correct MRS entry.

4.6.1.1.2 x PDSAP SI behind a PDMTP aware NAT without use of PDMTP adjacency In contrast to the previous subsection, interception by the PDMTP aware NAT could be handled without establishing any adjacencies or MAs terminating at the node provided that the PDMTP is able to inform the PDMTP SI of the translated embedded addresses and ports. This approach might have less real-time implications but requires some additions to the PDMTP protocol. This would necessitate additions to both the PDISC message and the PDISC-RESP message, whereby the PSI behind the NAT is informed (by the PSR co-hosting that specific NAT) of the translated data flow source address and transport port as well as the signaling message's translated source address.

The received PDISC-RESP message should not update the MRS entry's downstream node address. Instead the PSI retransmits its previous PDISC message with the updated translated addresses and ports.

We show this alternative in Figure 4.8. This approach without adjacencies requires (just as with the other method) that both the data flow source address and transport port as well as the signaling message source address be part of the PDMTP message. The intercepting node only interacts with the PDISC message when there is an inconsistency between the embedded data flow source address and transport port and the message source address, and their translated values. The PDISC-RESP message semantics should consider the above requirements. The main advantage of not using a PDMTP adjacency is the reduced latency due to avoiding

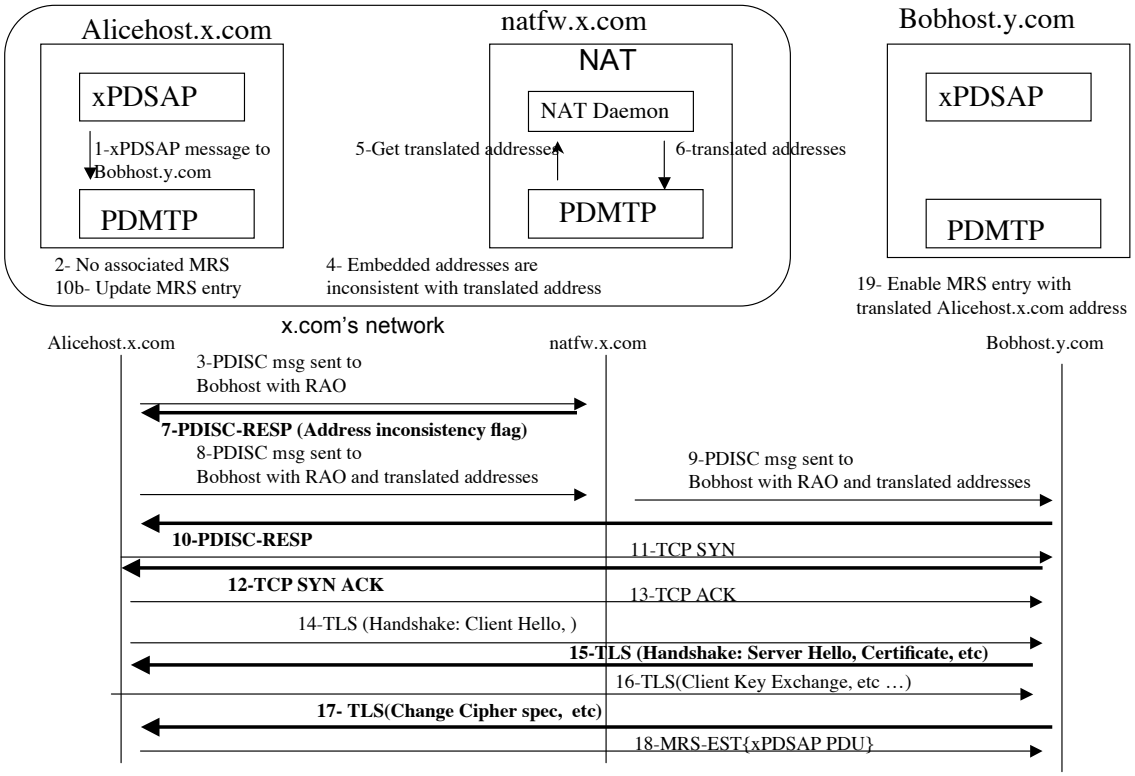


Figure 4.8: xPDSAP SI with a Downstream PDMTP Aware NAT with Bypass

the setup of the extra PDMTP adjacency and related MA establishment time. However a disadvantage to this proposal is that the PDMTP PDISC message creates a NAT bind without requiring a security association to be established (in which the identity of the PSI would be authenticated), this drawback could be used to mount DoS attacks on the node's memory and address and transport port pools (similar to DoS attacks potentially launched with STUN as discussed in 2.5.2).

We have preferred informing the PSI of the presence of a NAT (vs transparently modifying the PDISC addressing information) to avoid the issues that we have raised in Chapter 2 when ALGs are used (integrity protection, encrypted content, compatibility with protocol extensions, etc.).

Moreover to handle intra-realm issues discussed in Chapter 5, we shall use a NAT counter in the PDISC message informing the PSR (the one hosting an x PDSAP) that a PDMTP aware

NAT was on the path.

When sending its first PDISC message, the PSI sets the counter to 0. For cases where the SI is behind NATs associated with the network in which it is hosted, upon return of a PDISC-RESP with translated address values (or other hints indicating that a PDMTP aware NAT is responding), a new PDISC message is sent with the updated addressing information and with an incremented NAT counter value.

The determination of the message direction with regards to the private network serviced by the NAT (i.e. inbound or outbound message) is critical for handling NAT traversal. We assume that this network will have administrative responsibility for the NAT and will be able to configure it appropriately. Hence all implementations should have the ability to indicate to the PDMTP layer on an intercepting PDMTP aware NAT if the message was inbound or outbound with regards to the administrative private network.

4.6.1.1.3 x PDSAP SR behind a PDMTP aware NAT with use of PDMTP adjacency We have considered above the case of a "x" PDSAP SI, in this subsection we discuss the requirements and impacts when deploying a x PDSAP SR without a co-hosted PDNFS implementation while using the PDMTP adjacencies with the x PDSAP unaware NAT (but which is still a PDMTP aware NAT).

The first problem that we foresee is a difficulty with reaching the x PDSAP SR with messages sent by x PDSAP SIs outside the NAT as the SR's address is a private one which cannot be used from outside the address realm serviced by the NAT. We have discussed this issue in Chapter 3 and demonstrated that the best solution was for the PDSAP SI to send the PDSAP messages to the data flow receiver address and let the PDNFS aware NAT use the data flow receiver address and port to determine the translated SR's address.

There are two options that we discussed in Chapter 3 to allow an end-host to determine its translated address and transport port for a data flow recipient:

- for UDP transport STUN (introduced in Chapter 2) is an option
- the use of the PDNFS PDSAP (for all IP transport protocols) but this is not applicable in this scenario (since the x PDSAP SR is not co-hosting a PDNFS SR).

In this scenario we shall assume that the x PDSAP SI was aware by some method of the x PDSAP SR's address (for example STUN was used and then the user application client provided the data flow recipient address and UDP port via the application protocol).

Upon receiving a PDISC PDMTP message, the PDMTP SR (on the PDMTP aware NAT) analyzes the embedded PDSAP type although the PDSAP is not supported, the message will be processed as follows and is pictorially described in Figure 4.9:

- The PSR responds to the PDISC and follows the procedures discussed in section 4.3
- The PSR notifies the NAT daemon that a message was received from an unsupported PDSAP which requires embedded address and transport port translation (this is inherent to the PDSAP type). The x PDSAP message is sent to the upper layers (PDMTP API) and the PSI on the node is instantiated to send the x PDSAP downstream to the appropriate destination (translated address of the original data flow destination). The original data flow destination address and port is used by the NAT to identify the x PDSAP SR address for which the message is destined (requires NAT bind table lookup). This method as well as the alternative method described below could be exploited to launch flooding attacks on the SR node. We propose to use a correlation mechanism (nonce) that would be stored on the PDMTP aware NAT and communicated to the SI through the user application protocol. The correlation information could be the hash of the data recipient's address and port on the data receiver host interface (i.e., an address that would be locally scoped and not publicly registered). This correlation information is quite practical as the information would already be stored in the NAT bind table and this information could only be available to entities that were able to intercept user application messages (for example SIP messages). This correlation information would only be sent in PDISC messages when the SR is believed to be behind the NAT. User level applications could provide such hints by providing a list of address and ports on which application data flows could be received [73] [74]. We discuss this solution further in Chapter 5

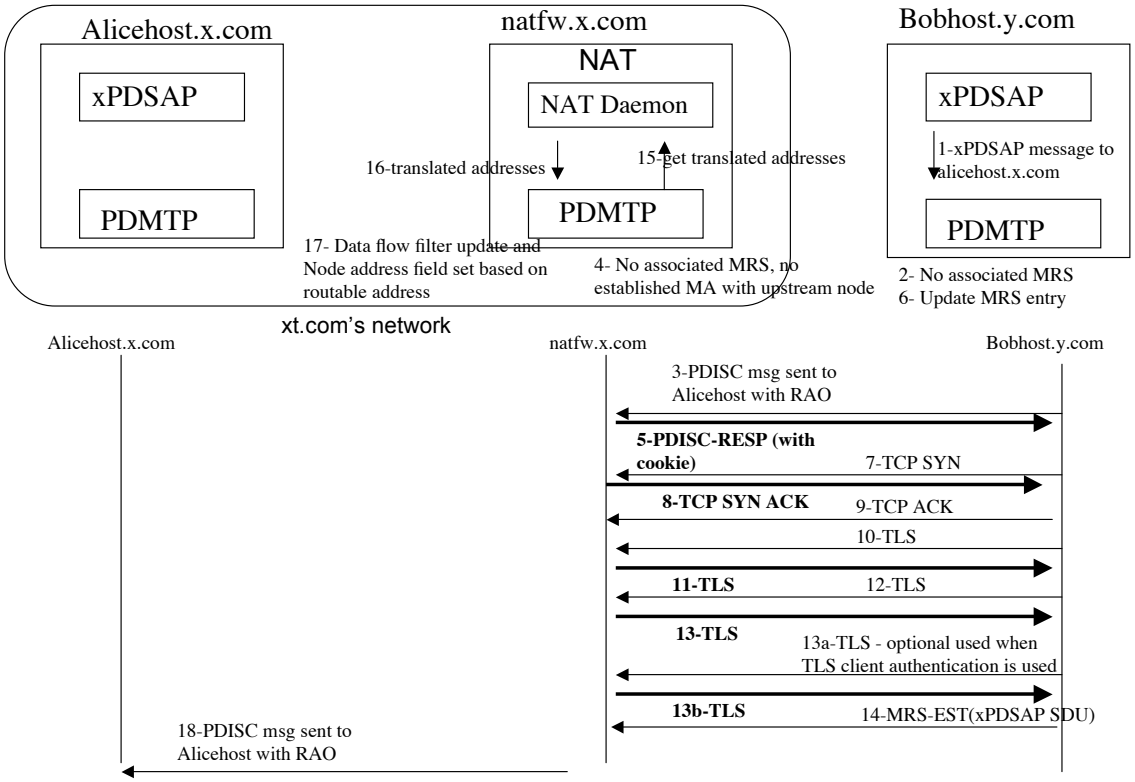


Figure 4.9: xPDSAP SR with a upstream PDMTP aware NAT

4.6.1.1.4 x PDSAP SR behind a PDMTP aware NAT without use of PDMTP adjacency The alternative to the above approach would be to continue intercepting messages but simply transparently redirecting (without responding to the PSI sending messages) them to the proper destination without changing the embedded addresses. Instead new addressing objects are added (containing the translated data flow destination address and transport port as well as the translated message destination address).

As opposed to the method used for SI's behind a NAT where an adjacency is not created, this method is transparent to the SI signaling the SR behind the NAT (i.e. the SI is not aware of the operation). This is required to avoid leakage of private addressing information. Instead of replacing the original embedded addressing information in the message we have preferred to keep the original address and add a new addressing object to allow the PSR to know that it is behind a PDMTP aware NAT (and know its translated address, i.e. the original destination

address of the message). The NAT count object introduced above is not incremented for inbound PDMTP messages.

In Figure 4.10, we provide detailed message sequences of the establishment of an IPsec Messaging Association while using IKEv1 for cryptographic parameter exchanges (Shared Key, cryptographic algorithm and aggressive mode). The message sequence analysis shows that the PDISC message is updated by the PSR instance (on the PDMTP aware NAT) with the translated data flow destination address and port as well as the signaling message destination address; in addition a new object provides the address and port needed to allow the PDMTP SI to establish TCP, SCTP and IKEv1/v2 communications. The latter addresses and ports are the translated address and port matching the assigned (and well known/static) TCP, SCTP, IKEv1/v2 ports on the xPDSAP SR. We shall discuss in more details IKE considerations for NAT traversal in section 4.6.4. The outcomes of this subsection and detailed message sequences

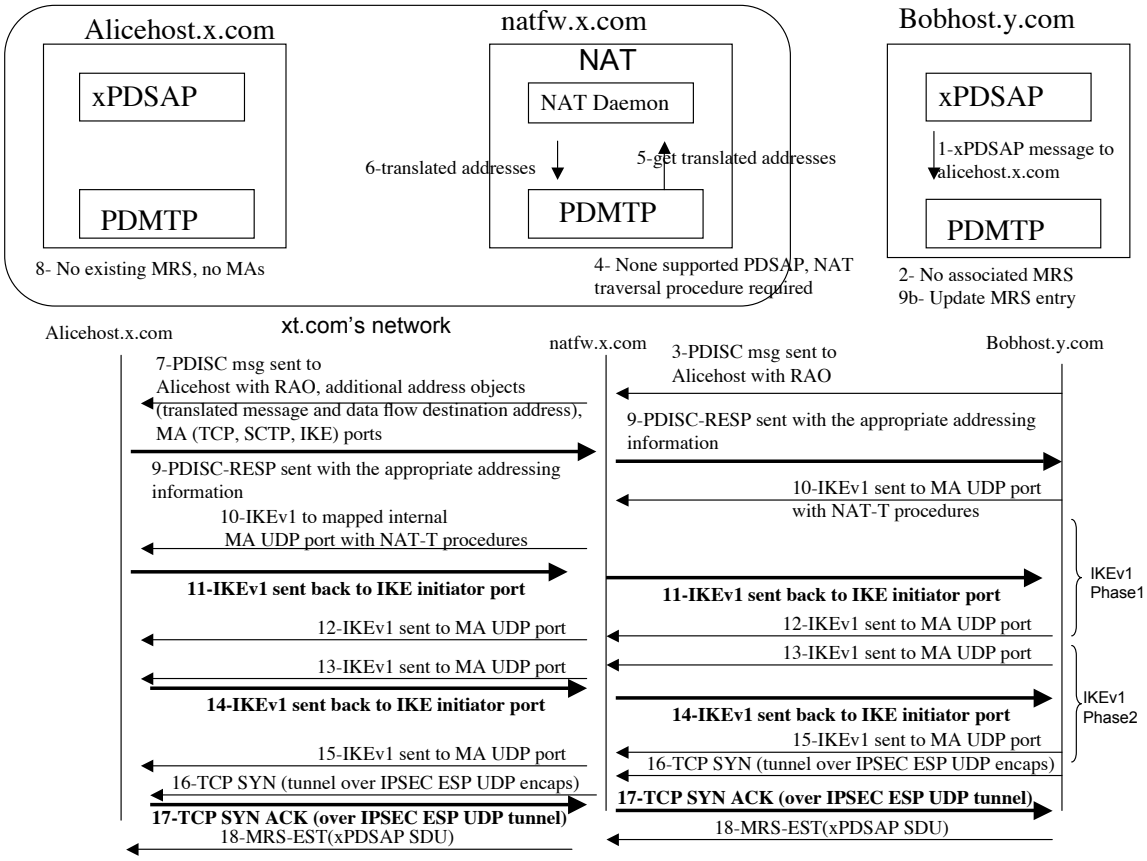


Figure 4.10: xPDSAP SR with a upstream PDMTP aware NAT with bypass

are that for real-time applications it would be better to avoid creating PDMTP adjacencies for relaying PDMTP messages (the ones not related to the PDNFS PDSAP if it was supported on the NAT node) across PDMTP aware NATs. The recommended approach would be to allow the PDMTP SR to learn the translated addresses for the data flow and signaling message source and destination as well as which appropriate transport ports to use for establishing Messaging Associations.

When the returned NAT count value is not zero, the PDMTP layer has to inform the PDSAP layer about the NAT count value; this behavior would allow the PDSAP layer to update its message when it includes the NAT count value as we shall see Chapter 5.

4.6.1.2 PDSP Node with a PDNFS PDSAP and with Downstream/Upstream PDNFS Aware NAT

When a PDSP supports the PDNFS PDSAP in conjunction with other PDSAPs, the node benefits from using the PDNFS PDSAP capabilities to learn the data flow's recipient address and port. The data flow recipient address would serve as the destination address of messages sent to the PDNFS SR (similarly to other PDSAPs SR as seen in the previous subsection). As opposed to other PDSAPs not supported on a PDNFS aware (and hence PDMTP aware) NAT, the PDNFS's PDMTP related messages should be handled by the PDNFS aware NAT by establishing PDMTP adjacencies.

In Chapter 5 we shall discuss the procedures used by the PDNFS protocol to reserve an address and port from its external address and port pools (in general that address is a public address unless the network contains cascaded NATs) bound to the private local address and port used by the data receiver. In this subsection we shall assume that the latter procedure is possible by establishing communications between the SR and PDNFS aware NAT and only focus on the potential issues that might be encountered.

Once a PDNFS SR has learned the translated data flow recipient's address and transport port number, that information will be communicated to the PDNFS SI typically through a co-hosted user application protocol agent (for example a SIP UA implementation [4]). Once the PDNFS SI is aware of the data flow recipient address and port it is able to send the message to that address and to the well known PDMTP port. The PDMTP operations will be similar

to the ones above in terms of learning to which address to forward the PDSAP message based on a NAT bind lookup by using the data flow destination address and transport port as input. Messages sent by a PDNFS PDSAP SI are handled in a similar way to x PDSAP messages sent by a x PDSAP SI as discussed in subsection 4.6.1.1.

4.6.2 PDMTP Interactions with Downstream/Upstream PDNFS/PDMTP Unaware NAT

We shall analyze these interactions while considering two scenarios: one where there is one PDMTP node behind the NAT and one where there are two or more.
PDMTP issues with PDMTP or PDNFS unaware NAT are very complicated to solve:

- Message forwarding is impossible if the data flow receiver address and port is used as a rendez-vous mechanism for sending PDSAP messages (encapsulated in PDMTP messages). There are two main options that we could consider:
  - Use alternative approaches to the PDNFS PDSAP to learn the address and port on which a PDSAP SR could receive messages, including STUN. However out of band mechanisms are required to provide the address and port on which a PDSAP SI/SR could receive messages; in practice this would require updates to user application protocol specifications which are not always simple (they require standardization processes, implementation changes and product upgrades).
  - Forward messages to the data flow receiver address and port (which would have been learned by alternative mechanisms such as STUN when applicable): this would require a sophisticated demultiplexing scheme based on the UDP datagram payload (or at least a small part of it at the beginning of the payload) to identify the PDMTP messages. One such mechanism is currently used by STUN as the same UDP socket is used to send/receive RTP and STUN datagrams; for example, assuming the data flow uses the RTP protocol, let the two most significant bits of the first 32 byte word of every signaling message be 00 which is not a valid possibility for RTP datagrams. Although this solution is known to work with RTP packets it may well not be

applicable to other data flow protocols. Hence the first option is more suitable for general usage.

- Embedded address and port translation:
  - Translation for data flow source and destination address and port: this problem could be solved by using STUN depending on STUN's applicability as discussed in section 2.5.2 and if NATs interleave with x PDSAP instances on the path then the mapped address and transport would only be applicable at the last NAT encountered. We show the applicability of STUN with interleaved PDMTP unaware NATs and xPDSAP instances in Figure 4.11.

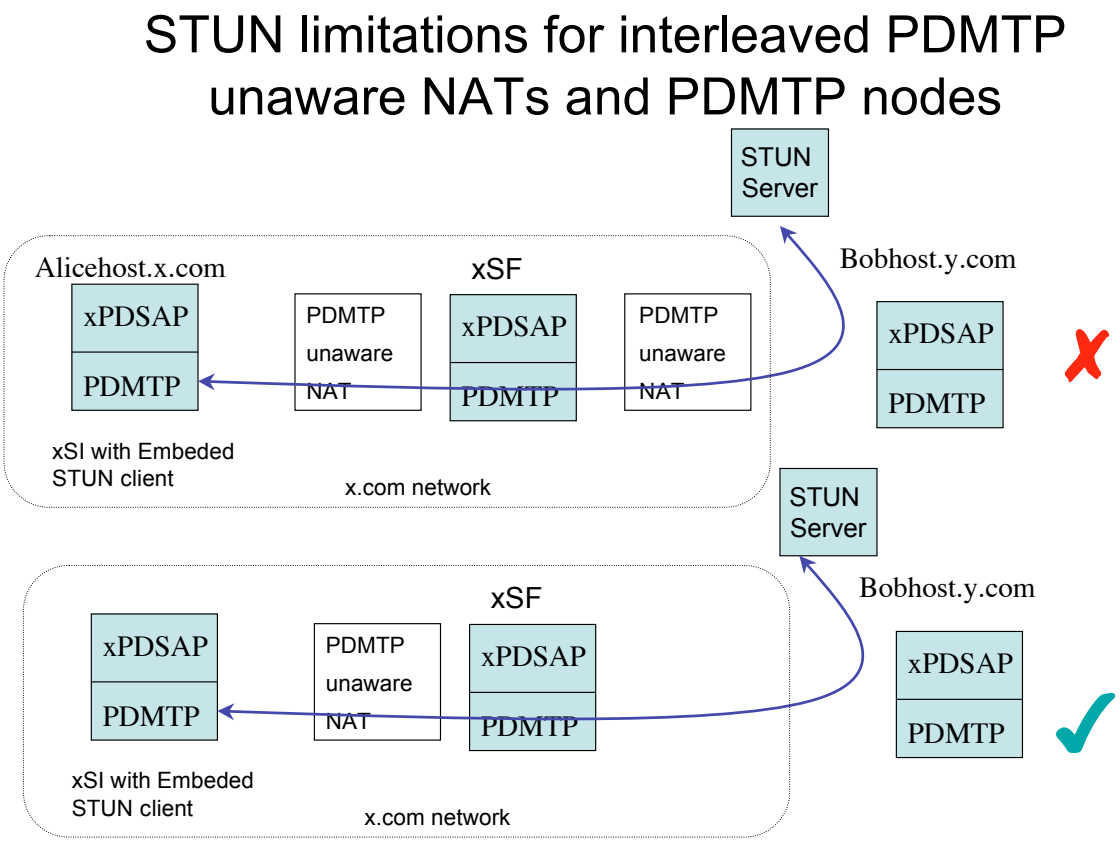


Figure 4.11: Applicability of STUN when used with interleaved PDMTP unaware NATs

  - PDMTP Message source and destination address: the reason for embedding the PDMTP message source address was to let the intercepting PSR determine if it

already has an established MRS between the PSI and the PSR by directly providing both the message source and destination address to the PDSAP layer. This issue cannot be solved by using the socket API to recover the real message source address as the PSI sending the PDMTP messages is spoofing the SI's address. However the use of STUN, when applicable (as shown in Figure 4.11) could potentially help in resolving this issue.

- Messaging Association address and port (TCP, SCTP and UDP for IKE) information: there is no practical solution[7] to solve MAs using TCP or SCTP since reflector methods are not applicable to TCP or SCTP (for example STUN is generally applicable only to UDP and this is further discussed in section 2.5.2). STUN could be used (depending on its applicability as discussed in 2.5.2) for determining where IKE messages should be sent when establishing an IPSEC MA using IKE. PDMT-RESP message semantic extensions discussed in subsection 4.6.1.1 would be applicable.

We have proposed several methods to allow, depending on the applicability of STUN usage, any PDSAP node to operate behind a PDNFS or PDMTP unaware NAT. However the only realistic approach to solving PDNFS or PDMTP unaware NAT traversal would require extensions to user application protocols. Depending on the practicability of these extensions PDSAP nodes might or might not be able to function in the presence of unaware NATs.

4.6.3 PDMTP Only Aware NATs vs PDNFS Aware NATs

We have seen in the earlier sections that a PDMTP only aware NAT allows messages to be forwarded properly while preserving the functionality of other PDSAPs (i.e., not the PDNFS). A direct consequence of the NAT traversal capabilities of the PDMTP layer is that one could question in that case the role of the PDNFS.

The PDMTP layer provides the translated address and port for the data flow as well as the signaling message translated address and port, however this functionality is only provided for

[7]MA establishment could be inverted by allowing the PDMTP instance behind the NAT to send the initial TCP or SCTP message however in many cases both PDMTP instances (PSI and PSR) might be behind PDNFS or PDMTP unaware NATs

one single PDMTP hop. In practice when more than one PDMTP only aware NAT separates an SI or SR hosting a xPDSAP implementation from the public network, the PDMTP layer does not provide the translated address and port information from the outermost NAT (the one at the edge of the public network). Hence when more than one NAT is used within either the SI or SR network the PDMTP layer on its own cannot solve the NAT traversal problem.
In addition the PDMTP layer on its own cannot solve the NAT traversal problem when the SR is behind a NAT. In that case external means have to be used to determine the SR's publicly routable address; we have proposed using STUN but this still has a number of limitations summarized here.

- STUN is only applicable to UDP, and
- it is vulnerable as a vehicle for Denial of Service attacks on memory and port pool resources.

4.6.4 IKEv1/v2 Considerations for PDSP Nodes behind NATs

There has been considerable work [75, 76] designed to solve the NAT traversal issues of IKEv1 and IPSEC, however [75] it appears to us that some difficulty remains related to the IKE UDP port change which is intended to occur after discovery of a NAT between the IKE initiator and the IKE responder.
According to [75], even if the IKE exchange started by sending the first message from the IKE initiator (PSI node in Figure 4.10) on UDP port 500[8] to a specific port (provided within the PDISC-RESP message), a port change is required; this would require an unnecessary message exchange to communicate the appropriate port. An alternative to the additional message exchange would be to provide two port numbers in the PDISC-RESP, however since this would still require a modification of the IKE stack we recommend using the same port number (instead of following the recommendation of [75], which is more applicable in traditional VPN remote access usage).

[8] A different port number would need to be used if there was more than one PSR behind the NAT

4.7 Firewall Traversal Implications on the PDMTP

As we have seen in Chapter 2, Firewalls impact network communications that are using traffic addressing patterns that are not listed in the list of filter matching expressions of allowed packet flows.

Firewalls may impact the PDSP protocol suite by preventing the forwarding of PDMTP messages during the following the phases:

- Adjacent neighbor discovery phase: the firewall could discard the UDP encapsulated PDISC, PDISC-RESP messages.
- Messaging Association establishment: the firewall could discard TCP, SCTP or IKE messages used to establish the MAs.
- Message transfers: if a network is multi-homed with more than one Firewall deployed to protect each of the network access connections, subsequent PDSAP messages or data flow messages could traverse a different firewall from the one traversed by the MA establishment packets. As a result the firewall traversed might discard messages sent on MAs established via one of the other firewalls or data flow messages for which SBS has been installed on another firewall.

We shall analyze the traversal of the PDMTP protocol message through PDMTP aware and unaware Firewalls.

4.7.1 Message Traversal of PDMTP Aware Firewalls

When a firewall is PDMTP aware, it could forward the discovery messages without establishing any PDMTP adjacency (when the relevant PDSAP is not supported on the firewall node). However this would allow attackers to attack the SR or SI that would be receiving these messages.

Since no adjacency is established and hence no secured MA is established with the assumed neighbor node, the assumed neighbor is therefore not authenticated which opens the door to

message flooding attacks. One possible solution to the flooding attack is to limit the neighbor to a maximum number or rate sending of discovery message but this mechanism is weak against address or identity spoofing attacks (the reverse routability check would require the establishment of an MA with the Firewall, which induces real-time impacts and would also need to be bound to the neighbor identity). Another mitigation method would be to use the same correlation material used for NAT traversal and discussed in section 4.6.1.1, however this might have issues as we shall see in Chapter 5. We can conclude that there are tradeoffs between flooding attacks mitigation and real-time impacts on applications.

4.7.1.1 Messaging Association Establishment and Message Transfer

If an MA is not to be established with the PDMTP aware firewall, establishment of MAs between PDMTP aware nodes on either side of the firewall requires that the firewall allows the forwarding of suitable transport protocols such as TCP or SCTP, and key exchange protocols such as IKEv1/v2 (encapsulated in UDP). As a consequence, the PDMTP aware firewall needs to intercept all PDISC and PDISC-RESP messages to determine the type of MA being proposed and on which transport ports the MA will be established. In addition the PDMTP aware Firewall would need to analyze IKE messages to determine the SPI[9] used and addressing information for the IPSEC SA (for IPSEC MAs).

4.7.2 PDMTP Unaware Firewall Traversal

PDMTP unaware Firewalls could be configured to allow packets known to encapsulate PDMTP messages however if the MA establishment and MA itself uses dynamically defined traffic filtering characteristics the Firewall will discard all these messages and no PDSAP messages could be exchanged.

[9]This is not always possible unless using a generic shared key for all PDMTP nodes and requires the establishment of the IKE Security Association

4.8 Summary of PDMTP NAT and Firewall related considerations

There are tradeoffs for both NAT and Firewall traversal of PDMTP messages: the primary consideration is whether to favor real-time performance over tighter security required for mitigating message flooding attacks (for firewalls) or memory/port pool exhaustion attacks (for NATs).

The real-time friendly solution bypasses MRS and MA establishment, whereas the better secured solution requires the establishment of both MRS and MA establishment, however the PDMTP layer would need to instantiate PSI or PSR instances on its own when the node does not support the PDSAP specific to the message being sent. In the remaining sections of this dissertation we have favored the real-time friendly solution.

4.9 Special Message Routing Treatments

As we mentioned in Chapter 3, the goals of the PDSP protocol suite are to be effective during the protocol migration phase and allow the protocol to operate in both path-coupled and path-decoupled modes. In this section we shall consider the impact of such requirements on the PDMTP layer.

4.9.1 Proxy Mode or the Last Downstream PDSP Hop Scenario

During the migration phase where not all Middleboxes or application end-hosts have been upgraded to support the PDSP protocol suite (the appropriate PDSAPs and the PDMTP), the supported PDSAP and the PDMTP should at least work in the network segments where this technology is available.

Since the PDSP protocol suite normally requires a message to be sent from the message source to the message destination, if the message destination does not support the specific PDSAP or the PDMTP, the PDSAP protocol should still be effective within the network segments that support it. This implies that when the last downstream PDSAP node is discovered, that node should notify the upstream PDSAP nodes on the path that there was no PDSAP SR and that it is acting as the last downstream PDSAP node (in essence a PDSAP proxy).

In certain cases this behavior could be even used at the edge of an administrative network.

In case a temporary link failure occurs during the discovery of some adjacent peer, it is quite probable that a downstream node could deduce that it is the last downstream node supporting the requested PDSAP. This problem could easily be detected by using the user application data flow connectivity loss detection mechanisms, which would detect that the application is misbehaving and would require a rediscovery of all the nodes on the path. Hence the ephemeral link failure would only induce a temporary condition.

4.9.2 Path-Decoupled Mode Of Operation

In Chapter 3 we have discussed the use of a Path-Decoupled mode of operation for the PDSAP protocols in cases where the application end-hosts have not yet been upgraded to support the PDSAP protocols and the PDMTP.

The use of the Path-Decoupled mode impacts the protocol due to the following requirements and the consequent protocol changes which they mandate:

- The message sender is not necessarily the data flow sender.
- The message sender may not necessarily want the intermediary nodes to process signaling messages if they are not the last downstream node supporting the required PDSAP or if they are not the downstream node at the edge of the administrative domain. To meet this requirement we propose to use a specific RAO value that instructs the PDMTP layer to only respond to the message if the message can't be forwarded.
- Only the destination node should process the signaling message. This would be equivalent to using the PDSAPs in a way that is close to targeted signaling protocols (section 3.3: the PDMTP discovery phase is changed by not inserting the RAO option. PDISC, PDISC-RESP and MRS-EST could still be used to negotiate MA types, MAs are established consequently and PDSAP messages established as if when the RAO is used.

We have discussed in section 4.3.1 the impacts on the PDSP protocols when the message source address is not the data flow source address, and concluded that these issues are unavoidable. Furthermore the second requirement is almost identical to the last downstream node discovery

requirement. This requirement could be met with the PDMTP protocol operations that we have defined in this chapter.

4.10 Threats Analysis

The PDMTP protocol could be subject to the following threats:

- Man In The Middle attacks (MITM):
  - A rogue network intermediary could claim to be a valid downstream or upstream PDMTP node and attack the PDSAP protocols and hence be able to mount attacks on the network by opening pinholes or redirecting traffic through fake NAT binds (in case the PDSAP is the PDNFS). We believe that this threat could be mitigated by mandating the use of security associations that require the peers to have credentials asserting their role in their specific network. We discuss this mitigation method in Chapter 6.
  - The rogue network intermediary could impact MRS states by inserting itself in the message path after the MRS was created on a node. This attack would basically seem to the nodes as a routing change.

    Proposed mitigation method: when a PDMTP discovers that its PDMTP upstream or downstream neighbor has changed for a specific MRI it SHOULD only update its MRS entry if the PDMTP neighbor successfully passed the establishment of a security association where its role in the network would be validated as discussed above.
  - Silent message changes: the on-path attacker could change PDMTP messages without being known by the PDMTP peers exchanging messages.

    Proposed mitigation method: all PDMTP adjacencies must be secured, hence all messages should be sent over security associations (i.e. the MTR should always require secure transfer). Messages sent without protection (i.e. PDISC and PDISC-RESP, when no MA is established), should be repeated over the MA to detect the potential changes made by the attacker.

  - Message dropping: an on-path attacker could drop messages. There are is solution to mitigate this attack.
  - Message redirection: the on-path attacker could redirect messages to new destinations causing potential message floods on both the original message source and the new destination selected by the attacker.

    There is no solution to the attack when the new message destination does not support the PDMTP protocol. In case the message destination supports the protocol, the message source will be able to detect that the destination is not the right one.

- Message Flooding attacks: these attacks will potentially lead to resource exhaustion either of memory or processing power. In certain cases the attacker would use spoofed addresses and the spoofed address's host would receive all the response messages.

4.11 Protocol Semantics

In the previous sections of this chapter we have covered the operations of the PDMTP protocol. In this section we shall discuss the semantics required for each of the PDMTP protocol messages. The PDMTP protocol operations require six different types of messages:

- PDISC: the PDMTP downstream neighbor discovery
- PDISC-RESP: the response message to the PDISC, this message would be sent by the discovered downstream peer
- MRS-EST: this message is sent by the PDMTP upstream neighbor with or without MA establishment (if no specific MTR was provided)
- ERROR: this message is sent by the PSI or PSR to notify errors
- PDATA: this message carries the PDSAP messages
- HELLO: this message is used to refresh established Messaging Associations

All messages have in common the following contents: message type, node identity and message direction (useful for messages sent in either upstream or downstream).

4.11.1 PDISC Semantics

The PDISC message carries the following information:

- The PSI identity: this field combined with the PSI(s) address should be unique
- The PSI address(es): one IPv4 address or/and one IPv6 address.
- The message direction bit: set to upstream (provides the message direction to the receiver entity)
- MRSUID: the unique identity for the MRS entry
- The Messaging Association type (or stack) proposal: TCP, SCTP, TLS, DTLS, TCP/IPSEC with IKEv1 or TCP/IPSEC with IKEv2. The proposal includes a list of Messaging Association types with each Messaging Association type listing the protocols used (TCP, SCTP, TLS, DTLS, IKEv1/IKEv2 with IPSEC).
- Data flow filter descriptor providing some or all of:
  - IP Protocol type (for example: TCP, SCTP, UDP, DCCP, IPSEC (if the transported PDU will be protected with IPSEC))
  - Data flow source address
  - Data flow source demultiplexer: source transport port
  - Data flow destination address
  - Data flow destination demultiplexer: destination transport port or SPI (for IPSEC)
  - Data flow label (for IPv6 flows only)
- PDMTP-I cookie
- NAT count: a value initially set to 0 when the PDISC is sent for the first time
- A correlation token: this field is a quantity computed based on information sent through the user application (as discussed in sections 4.6 and 4.7)

4.11.2 PDISC-RESP Semantics

The PDISC-RESP semantics carries the following information

- The PSR identity: this field combined with the PSR address should be unique
- The PSR address(es): one IPv4 address or/and one IPv6 address
- The message direction bit: set to downstream (provides the message direction to the receiver entity
- MRSUID: the unique identity for the MRS entry
- The accepted Messaging Association type (or stack) proposal: this is used to inform the PSI of the selected mutually acceptable MA type (and the required keep-alive period) supported by both the PSI and PSR. In addition, the PSR provides the address and port to use for establishing the Messaging Association for each acceptable type (the port may be different than the well known MA protocol'S port number).
- Data flow filter descriptor providing some or all of:
  - IP Protocol type (for example: TCP, SCTP, UDP, DCCP, IPSEC (if the transported PDU is protected with IPSEC))
  - Data flow source address
  - Data flow source demultiplexer: source transport port
  - Data flow destination address
  - Data flow destination demultiplexer: destination transport port or SPI (for IPSEC)
  - Data flow flow label (for IPv6 flows only)
- Translated MA recipient address and port: this is only sent when a PDMTP aware NAT is on the path between an SI and SR which does not support the specific PDSAP that caused the PDISC message to be sent (this is discussed in section 4.6.1.1)
- Translated Data flow filter descriptor: this is only sent when a PDMTP aware NAT is on the path between an SI and SR which does not support the specific PDSAP that caused

the PDISC message to be sent (this is discussed in section 4.6.1.1). By its presence the PSI receiving the PDISC-RESP will be triggered to take into account the message without creating a new MRS entry (and to increment the NAT count object).

  - IP Protocol type (for example: TCP, SCTP, UDP, DCCP, IPSEC (if the transported PDU was protected with IPSEC))
  - Data flow source address
  - Data flow source demultiplexer: source transport port
  - Data flow destination address
  - Data flow destination demultiplexer: destination transport port or SPI (for IPSEC)
  - Data flow label (for IPv6 flows only)

- PDMTP-R cookie
- PDMTP-I cookie

4.11.3 MRS-EST Semantics

The MRS-EST carries the following information:

- The PSI identity
- The MRSUID
- The message direction bit: set to upstream (provides the message direction to the receiver entity)
- Original PSI proposal MA types list, this is required to detect MA downgrade attacks that might happen during the discovery phase since the PDISC message was not protected.
- PDSAP message
  - PDSAP type
  - Payload length

  - PDSAP payload

- PDMTP-R cookie

4.11.4 ERROR Semantics

The ERROR message carries the following information:

- The PSI or PSR identity
- The MRSUID
- The message direction bit: set to upstream or downstream (provides the message direction to the message receiver entity)
- Error objects: a list of error objects could be carried by the ERROR message
  - Error type: errors could be of the following types, administrative (for example node or user is not authorized to establish an MA), system failures (no more available memory or processing power) and protocol errors
  - Temporal effect: this information informs the message receiver if it is worthwhile to retry or not, two values should be provided such as a permanent error (would be the case of certain administrative and protocol errors) and an ephemeral error condition (could be potentially the case of a system failure).
  - Object length
  - Object value

4.11.5 PDATA Semantics

The PDATA message embeds a PDSAP PDU and should carry the following information:

- The PSI or PSR identity (identity of the message's source)
- The MRSUID
- The message direction bit: set to upstream or downstream (provides the message direction to the message receiver entity)

- PDSAP message
  - PDSAP type
  - Payload length
  - PDSAP payload

4.11.6 HELLO Semantics

The HELLO message's distinguishes itself from the other messages by using a special, predetermined MRSUID as the message is not specific to a MRS but to all MRS between two PDMTP peers on the MA that carries it.

- MRSUID: the protocol specification should use a special MRSUID value, in general such values use an "all 1" value
- The PSI identity: this field combined with the PSI(s) address should be unique
- The PSI address(es): one IPv4 address or/and one IPv6 address.

4.12 Summary

In this chapter we have described the PDMTP protocol operations covering PDMTP peer discovery, message routing and maintenance of PDMTP peering adjacencies. We have covered the NAT and Firewall traversal aspects of the protocol as well as the protocol semantics. The proposed protocol would allow the transmission of PDSAP messages between application end-point supporting the protocol as well as when one or none of the application end-points in communication supports the protocol.
The detailed protocol specification could be inspired by the GIST protocol specification [17] defined within the IETF, with some deviations to cover the required semantics of PDMTP (mainly fate sharing and NAT traversal).

Chapter 5

Path-Directed NAT and Firewall Signaling (PDNFS)

In this chapter we shall describe the PDNFS PDSAP that would be used to signal NATs and Firewalls to enable application data traversal.
The PDNFS PDSAP requirements can be inferred from Chapters 3 and 4. We shall summarize all these requirements and determine the appropriate mode(s) of operation for the protocol and the protocol semantics.

5.1 Terminology

- Edge-NAT: An edge-NAT is a NAT device that is reachable from the Internet and that has a globally routable IP address on the Internet facing side. An edge-NAT is deployed at the boundaries of an administrative domain's address realm.
- Edge-Firewall: An edge-Firewall is a Firewall device that is located on the demarcation line of an administrative domain.

5.2 The Path-Directed NAT and Firewall Signaling protocol requirements

The PDNFS protocol should meet the following requirements:

- Support of different deployment models including:
  - Only one PDNFS application end-host is behind one (or more) PDNFS aware NAT(s)

  - Only one PDNFS application end-host is behind one (or more) PDNFS aware Firewall(s)
  - Only one PDNFS application end-host is behind one (or more) PDNFS aware NAT(s) and Firewall(s)
  - Both PDNFS application end-hosts are behind one (or more) PDNFS aware NAT(s) or Firewall(s) (with all possible combinations)
  - Support of the appropriate semantics to handle the creation, the maintenance and deletion of NAT binds
  - Support of the appropriate semantics to handle the creation, the maintenance and deletion of Firewall pinholes

- Extensibility to handle new IP transport protocols
- The protocol should not create new security threats to the network infrastructure
- The protocol should allow the use of alternative NAT and Firewall traversal mechanisms in the network
- The protocol should work even when signaling messages are not forwarded beyond an administrative network's perimeter

5.3 One vs two separate protocols for NAT and Firewall signaling

Before going further in our analysis to meet the PDNFS' requirements we need to decide whether one or two separate protocols should be used to control the two different sorts of Middlebox - NATs and Firewalls. We shall base our reasoning on the required protocol semantics, performance and required interactions with the NAT and Firewall daemons, and the authorization policy database.

As a prerequisite to the single or separate protocol analysis, we will make a careful examination of NAT and Firewall signaling protocol semantics.

5.3.1 NAT signaling protocol semantics

A NAT signaling protocol should have the ability to create a NAT bind entry, maintain it and delete it. The type of NAT addressed by these semantics is a NAT that only handles address and, maybe, port translation which is the original purpose of NATs, filtering capabilities are excluded and handled by a Firewall daemon.
The purpose of using such a protocol is to allow an application end-host to receive application data flow packets as well as to receive PDSAP messages encapsulated in PDMTP messages from a data sender through one or more NATs which may be translating addresses either for the network hosting the sender or the receiver.

Under some circumstances, the data receiver does not necessarily know the data flow sender address, but needs to signal the edge-NAT to reserve addresses so that subsequent signaling from the data sender can reach the data receiver via the NAT and allow the application to learn these addresses so that it can inform the data sender. The NAT PDSAP messages used for this purpose from the data receiver are sent to an Opportunistic Address (**OA**). The main property of the OA is that the NAT PDSAP messages have to traverse the outermost NAT of the domain and create a binding which is allocated a registered address on its public side (i.e., in an edge-NAT). Moreover since these NAT PDSAP messages are not sent by the data flow sender, it might not be possible to guarantee that the associated data flow packets would go through the same Middleboxes as the ones traversed by NAT PDSAP messages; however since all packets from the data sender would be sent to the NAT (as it hosts the registered address provided to the application end-host sender) data flow packets **will** get the appropriate treatment.
Consequently the following messages with their associated semantics are required for such a protocol:

- A NAT bind creation message providing the data flow's source and destination address and transport port pairs (the identifiers of the sending and receiving sockets, i.e. with a local scoped private address for the receiving socket) providing the information that would be bound to the NAT bind entry requested in the NAT bind table. The destination address and port pair is the pair that will be bound to a counterpart address and port

pair selected by the NAT daemon. The address would be selected from an available pool of addresses and the port would be selected from an available pool of ports. If the NAT daemon was configured to perform twice NAT translations [2], the source address and port pair would also be bound to an address and port pair selected in the same way (but from different address and port pools) as the pair bound to the destination address and port pair. Additional information could be provided such as:

  - NAT bind lifetime: in case no NAT signaling messages are received by the NAT during the NAT bind lifetime, the NAT bind will expire and be removed from the NAT bind table. This behavior is different from that of traditional NAT bind entries where arrival of data flow packets could refresh the NAT bind entry. This new behavior offers a better network protection and simpler application behavior (as keep-alives are no longer required to be sent at the NAT bind timer pace) by the application.
  - Number of contiguous ports and port parity preservation: in case multiple data flows using the same address with contiguous source ports are used (this was typically the case of RTP and RTCP flows prior to [77]).
  - A specific translated (or mapped) address or address and port pair to be selected from the available list of address and port pools
  - Nonce to use for allowing message forwarding: if this nonce is sent, then the default nonce is not used (this was introduced in section 4.6.1.1). PDMTP messages used to encapsulate any PDSAP messages should only be forwarded (depending on the security policies of the network) if PDMTP messages have the appropriate nonce matching the NAT bind entry used to route the PDMTP message. Although this information is carried at the PDSAP layer (for end-to-end delivery), it is more appropriate to use it at the PDMTP level to enforce security checks prior to establishment of Messaging Associations.

- A message to refresh an existing NAT bind entry (there is no reason to prevent the implemented protocol using the same protocol message as that used to create a NAT

bind). The NAT PDSAP SF should have the appropriate information to validate that the message originator is authorized to perform this action.

- A message to update (including the removal of) an existing NAT bind entry, the NAT PDSAP SF should have the appropriate information to validate that the message originator is authorized to perform this action. There is no reason to prevent the implemented protocol using the same protocol message as the one used to create a NAT bind.

- An asynchronous event notification message sent by the NAT to alert the NAT PDSAP initiator of specific issues that would impact the forwarding of data or PDSAP packet flows. Notified events are events occurring due to hardware or software failures (including kernel panics) or planned outages (platform software upgrades or other). We do not foresee any specific events due to NAT bind management considerations.

- A response message indicating a success or an error condition (authorization, system errors and protocol errors) to one of the above messages (except the asynchronous event notification) as well as additional information such as the translated address and port pair(s) (for successful responses). Depending on the error type, an error condition duration time should be provided to inform the message sender if it is worth while retrying its request. Although the NAT PDSAP protocol could use a reliable messaging transport protocol, to ensure that response message matches a specific request message, a message sequence number should be sent in all the above messages and mirrored in their response messages.

  Specific errors which apply to the NAT PDSAP such as protocol errors and errors related to NAT bind creation and management:

  - Requested translated address or address and port pair is unavailable or already assigned
  - Exhausted port pool

Additional requirements on all messages are:

- Ability to authenticate the source of a message or objects within a message

- Ability to protect the integrity of a message or objects within a message sent by the original message source

5.3.2 Firewall signaling protocol semantics

A Firewall signaling protocol should have the ability to create, maintain and remove pinhole entries. This protocol is designed to allow a data flow sender and data flow receiver to request that Firewalls on the path between them permit the traversal of their exchanged packets.

When an application aware end-host signals a Firewall to establish a filter that will accept its inbound PDSAP packets so that it subsequently act as a data receiver, the PDSAP messages sent to the Firewall are not guaranteed to traverse the same Firewall(s) as either the inbound PDSAP message or the subsequent data flows since these messages will not be sent to the Firewall(s) addresses (in contrast to NAT PDSAP messages which are sent to addresses hosted by NATs). Hence the protocol might not always be effective in establishing the required state when used by a data flow receiver. Potential solutions to this problem are discussed in Chapter 6.

Based on the above, the following protocol messages with their associated semantics are required:

- A message to create a pinhole for a data flow identified by a flexible filter matching expression. This message should provide the requested lifetime for the pinhole. If no Firewall PDSAP messages are sent to the Firewall during this period, the pinhole will be removed from the pinhole table. Pinholes governed by the Firewall PDSAP cannot be refreshed by data flow activity (unless required by the network's security policies). As discussed in Chapter 4 all embedded addressing information would be part of the PDMTP message but be provided by the firewall PDSAP via the PDMTP$<->$FW PDSAP API. Additional information could be sent in this message such as:
  - the nonce to use for forwarding the message (as discussed in the NAT subsection), the nonce is provided to the PDMTP layer which would use it. The FW PDSAP would only carry the nonce end-to-end (vs hop-by-hop as operated by the PDMTP).

  - Specifications of additional data flows that might have the same source or destination addresses but with a contiguous port number. This is important when supporting deployed legacy implementations of RTP and RTCP not supporting the recommendations of [77].

- A message to refresh an existing Firewall pinhole entry (there is no reason to prevent the implemented protocol using the same protocol message as is used to create a Firewall pinhole): The Firewall PDSAP SF should have the appropriate information to validate that the message originator is authorized to perform this action.

- A message to update (including the removal of) an existing pinhole entry: the Firewall PDSAP SF should have the appropriate information to validate that the message originator is authorized to perform this action. There is no reason to prevent the implemented protocol using the same protocol message as the one used to create a pinhole.

- A response message indicating a success or error condition (authorization, system errors and protocol errors) to one of the above messages. Every error condition should indicate its temporal effect to inform the message sender if it is worthwhile retrying its request. Firewall signaling specific errors include protocol and pinhole lifetime errors. Although the Firewall PDSAP protocol could use a reliable messaging transport protocol, to ensure that response message matches a specific request message, a message sequence number should be sent in all the above messages and mirrored in their response messages.

- An asynchronous event notification message sent by the Firewall to alert the FW PDSAP initiator of specific issues that would impact the forwarding of data or PDSAP packet flows. Notified events are events occurring due to hardware and software failures (including kernel panics), and planned outages (platform software upgrades and others). We do not foresee any specific events due to pinhole management considerations.

Additional requirements on all messages are:

- Ability to authenticate the source of a message or objects within a message.

- Ability to protect the integrity of a message or objects within a message sent by the original message source.

5.3.3 NAT and Firewall Implementation Variants Considerations

In Chapter 2 we have discussed the various NAT implementations and Firewall implementations that are in use currently. This thesis aims to show how signaling can be used to eliminate the need for NATs and Firewalls to be aware of the user applications that are sending traffic through them, hence the analysis covers only NAT and Firewall implementations that do not include extensions that would make them user application aware (e.g., we do not consider boxes that contain Application Layer Gateways).
In Chapter 2 we have described several types of NATs, according to a classification that we have introduced:

- End-Point Independent Mapping NATs (EPIM): some of these implementations have filtering capabilities which makes these implementation a hybrid Network address translator and Firewall implementation.

- End-Point Dependent Mapping NATs (EPDM): this implementation variant also has filtering capabilities. In addition this type of implementation is expected to be rarely deployed based on recommendations of [41]

The mix of Firewall and NAT capabilities within one implementation box will greatly influence the NAT and Firewall signaling framework as we shall see in the following sections.

5.3.4 Analysis of a Solution using Two Different Protocols

This analysis will consider three deployment scenarios, shown in Figure 5.1, where NATs and Firewalls are deployed in different sequences as well as cases where the same node supports both NAT and Firewall capabilities. The following conventions are used for message naming in Figure 5.1:

- FIx: FW (Firewall) PDSAP messages exchanged between the FW PDSAP Initiator hosted within Netx (that initiates the exchange) and a FW PDSAP Responder external to Netx, where x could be 1, 2 or 3 as shown in the figure

- RFIx: FW PDSAP messages exchanged between a FW Initiator external to Netx (that

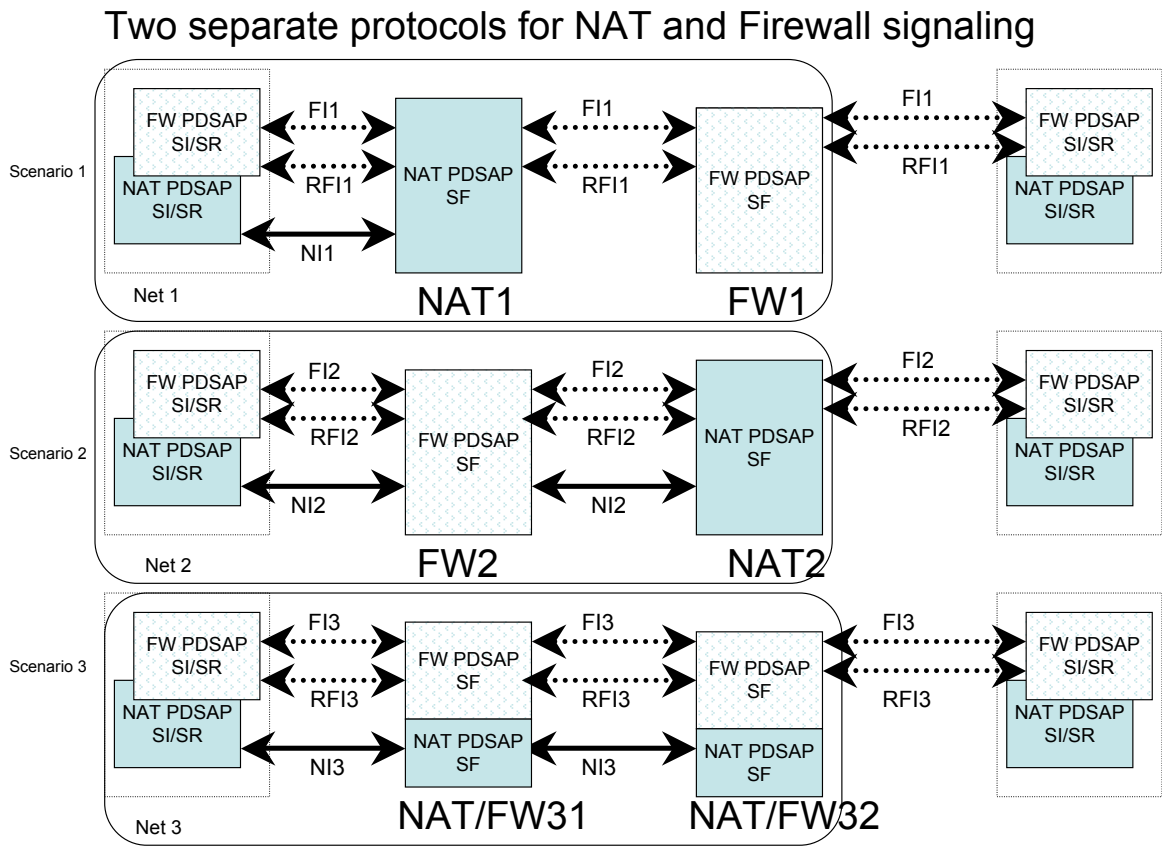


Figure 5.1: Analysis of two separate NAT and Firewall signaling protocols in diverse NAT and Firewall traversal sequences

initiates the exchange) and to a FW PDSAP Responder hosted within Netx, where x could be 1, 2 or 3 as shown in the figure

- NIx: NAT PDSAP messages exchanged between a NAT PDSAP Initiator (that initiates the exchange) and a NAT PDSAP Responder on the edge-NAT, where x could be 1, 2 or 3 as shown in the figure

5.3.4.1 Scenario 1 Deployment Analysis

In Figure 5.1, when traversing the NAT PDSAP aware NAT (which is implicitly PDMTP aware) FW PDSAP messages are forwarded downstream successfully. Due to the PDMTP NAT handling procedures discussed in Chapter 4, the data flow source and destination addresses embedded in the PDMTP header are properly translated. The main purpose of the NAT

PDSAP, in this network topology, is to create a NAT bind in order to have an address and port to provide to data sender application hosts outside the network serviced by NAT to allow them to send data traffic (as well as to send PDSAP messages) to the end-host behind the NAT (this is done prior to the FW PDSAP message exchange).

5.3.4.2 Scenario 2 Deployment Analysis

In scenario 2, the sequence of traversal is changed and NAT PDSAP messages traverse the FW PDSAP aware Firewall, and are then forwarded until reaching the edge-NAT. The Firewall forwards PDMTP messages encapsulating the NAT PDSAP messages without establishing a PDMTP adjacency (i.e., creation of a MRS entry).
The NAT PDSAP messages are used to create a NAT bind to receive user application data flow packets. The same NAT bind will be used to determine the address to use when forwarding PDSAP messages to the Net2's FW PDSAP SR. FW PDSAP messages are sent end-to-end up to the FW PDSAP SR.

5.3.4.3 Scenario 3 Deployment Analysis

In scenario 3, since every node in Net3 hosts both the NAT and FW PDSAPs the PDMTP layer will create adjacencies and the associated MRS entries. Messages will be forwarded as is normally done on a node supporting a PDSAP associated with intercepted PDMTP messages. When several NATs are deployed in cascade, the NI3 messages returned to the NAT PDSAP Initiator should provide all the translated addresses that the Initiator's host could have on each NAT on the path. This would avoid both:

- connectivity issues with certain devices (where only NAT/FW31 would be on the path), and
- wasting bandwidth resources due to back-hauling data flows up to NAT/FW32 when communicating with the previous devices hosted in a common addressing realm.

5.3.4.4 Interactions between the NAT, FW PDSAPs and the NAT and Firewall Daemons

In the previous subsections we have analyzed three different deployment options including one deployment using co-hosted NAT and FW PDSAPs. In Figure 5.2 we show the interactions between the PDMTP, FW PDSAP, NAT PDSAP, NAT daemon and Firewall daemon; we shall discuss API requirements in Chapter 7 and hence will not describe them in this chapter. As discussed in section 5.3.3, certain NAT implementations include packet filtering capabilities

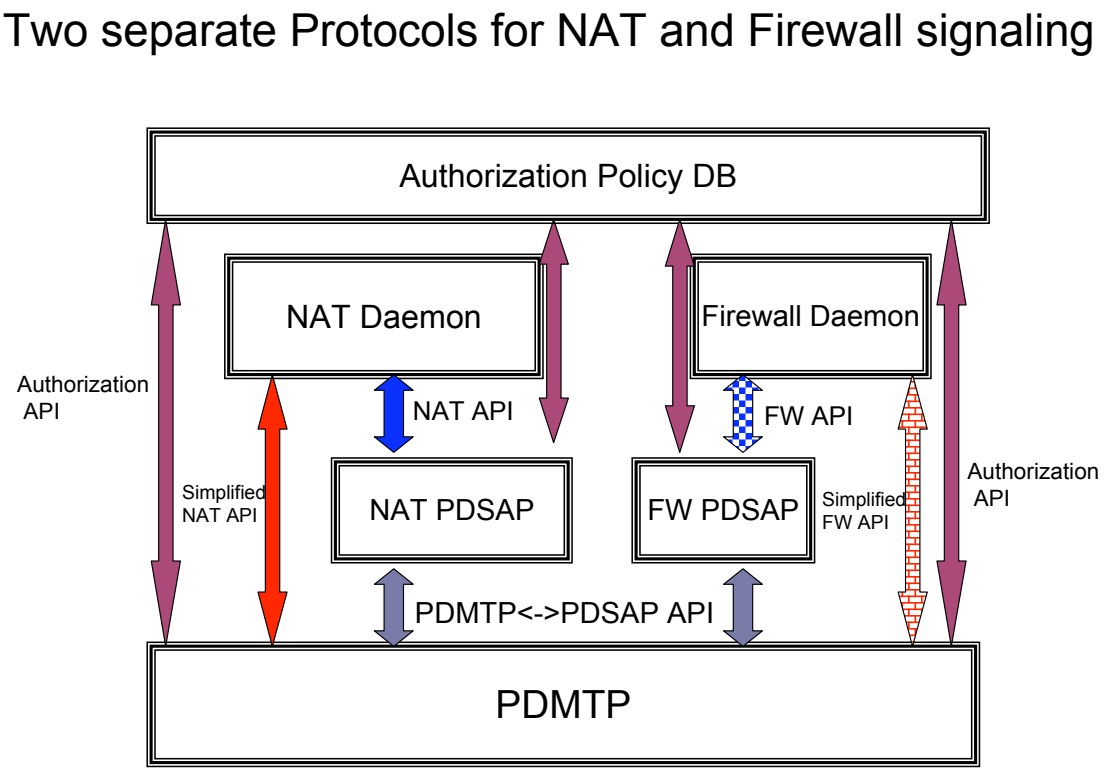


Figure 5.2: Interactions between all the PDSP layers and the NAT and Firewall daemons

which implies that the NAT implementation should also interact with the FW PDSAP to enforce restrictions on inbound data flow packet sources.

If the creation of a NAT bind was refused by the NAT PDSAP on the node, the FW PDSAP would not be aware of the failed request and hence a pinhole would have uselessly been created unless there is appropriate interaction between the two PDSAPs.

Similarly the Firewall pinhole creation might fail while a NAT bind was created. In that case there is no reason to retain the NAT bind. These two error cases could be handled by having the PDSAPs for the NAT and FW capabilities communicate with each other, however this inter PDSAP communication would significantly complicate the implementations of the PDSAPs. These interactions raise serious issues about the advisability of a model using two separate protocols.

5.3.5 Analysis of the Combined Approach where One Protocol is Used for Both Firewall and NAT Signaling

In the combined approach, the protocol semantics are not a simple merge of the semantics of the separate protocols for the following reasons:

- Correlating protocol states: for example, if a NAT bind creation has already been signaled and an incoming pinhole reservation is received on the combined NAT and Firewall Middlebox for the same data flow (as identified by the MRI), how should the PDSAPs correlate the two states as they were induced by messages sent with different MRSUIDs and have different MRS entries in the PDMTP layer.

- Two different classes of Signaling Initiators when dealing with combined middleboxes at the edge of the data receiver's network:

  - NAT bind creation, maintenance (refresh and update), and deletion messages will be sent by the PDNFS entity hosting the data receiver (or acting on its behalf)

  - Pinhole creation, maintenance (refresh and update), and deletion messages will be sent by the PDNFS hosting the data sender (or acting on its behalf)

- Fate sharing conditions occur when a Firewall and a NAT are co-hosted. In this case should a NAT bind refresh message be sent by the node hosting the data receiver (or acting on its behalf) or a Firewall refresh message be sent by the node hosting the data sender (or acting on its behalf); with two separate protocols, two messages are sent.

- Since the protocol should handle all NAT implementation types, pinhole creation messages

are interpreted on the NAT implementation as the configuration of the authorized data flow filter description. Hence in that case, should a NAT bind refresh messages be sent (by the node hosting the data receiver) or should pinhole refresh messages be sent.

We believe that these changes to the semantics could be easily addressed and will be discussed in section 5.9. We believe that the combined approach could result in a lower refresh (although this might not be perceptible in low duration interactions) message volume than the split approach since refreshing a pinhole (to allow traffic to a recipient) would refresh the NAT bind used to forward packets to the same data recipient. To complete the comparison between the two protocol frameworks, we revisit the three deployment scenario options in Figure 5.3. The

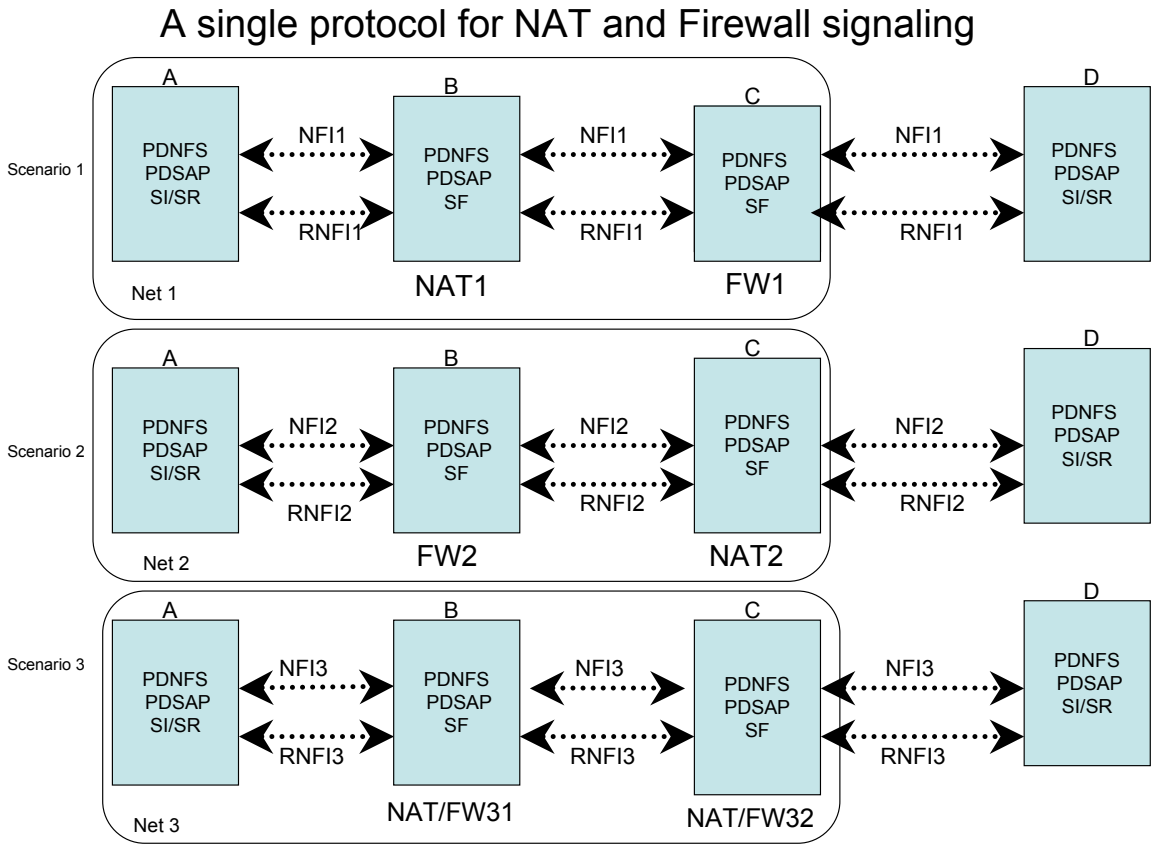


Figure 5.3: Analysis of a single protocol for NAT and Firewall signaling in diverse NAT and Firewall traversal sequences

following conventions are used for message naming in Figure 5.3:

- NFIx: PDNFS PDSAP messages exchanged (but initiated) by the PDNFS PDSAP Ini-

tiator hosted within Netx and a PDNFS PDSAP Responder external to Netx, where x could be 1, 2 or 3 as shown in the figure

- RNFIx: PDNFS PDSAP messages exchanged between (but initiated) by a PDNFS Initiator external to Netx and to a PDNFS PDSAP Responder hosted within Netx, where x could be 1, 2 or 3 as shown in the figure

5.3.5.1 Scenario 1 Deployment Analysis

In this scenario, a NAT bind creation message is sent by node A to the Opportunistic Address (defined in 5.3.1). Since node B is the edge-NAT for Net1 the message is no longer forwarded. Node A's translated address is provided to node D via a user level application protocol (for example SIP) to allow node D to send its PDNFS messages (RNF1) to node A. RNF1 messages allow the update of the previously created NAT bind entry (if the NAT daemon has entries with the associated external end-hosts). The matching of the NAT bind entry's with the RNF1 messages is based on the nonce (that we have discussed in section 4.6.1.1) and node A's mapped address and port included in the PDMTP's header's MRI. Similarly node A sends pinhole creation, maintenance and deletion messages to node D. These messages get proper treatment by node B and C, and are forwarded based on the PDMTP message's forwarding rules when an intercepting node supports the PDSAP with appropriate handling when the node supports NAT capabilities (as discussed in section 4.6.1.1).

5.3.5.2 Scenario 2 Deployment Analysis

In scenario 2, a NAT bind creation message is sent by node A to the Opportunistic Address, the message traverses node B however since there is no NAT on node B there is no point of establishing a PDNFS adjacency (and hence PDMTP adjacency for that message) the message is forwarded as if node B did not support the PDNFS PDSAP. In section 4.3.2 we have discussed how a Router Alert Option could be set to avoid establishing adjacencies prior to analyzing PDSAP message payloads; this will be used in this scenario. The NAT bind creation message is accordingly forwarded down the path reaching node C. Since node C is the Net1's edge-NAT it will respond back to Node A's NAT bind creation request. NFI2 and RNFI2 messages will

be handled in a similar way to NFI1 and RNFI1 messages when traversing firewalls and NATs.

5.3.5.3 Scenario 3 Deployment Analysis

In scenario 3, message routing adjacencies are established on nodes B and C for all messages. All the message forwarding is handled as defined in Chapter 4 when a PDSAP aware node receives its supported PDSAP messages. Message forwarding of Firewall pinhole requests through a NAT is similar to scenario 1 and 2 with the nonce usage.

5.3.5.4 Interactions between the PDNFS PDSAP and the NAT and Firewall daemons

The interactions between the PDSP layers (PDMTP and PDNFS PDSAP) and the NAT and Firewall daemons are a little bit different since the protocol stack is different as shown in Figure 5.4. In addition the error handling issues that we have discussed in section 5.3.4 are no longer applicable since errors are returned in the same protocol and hence event correlation is simpler.

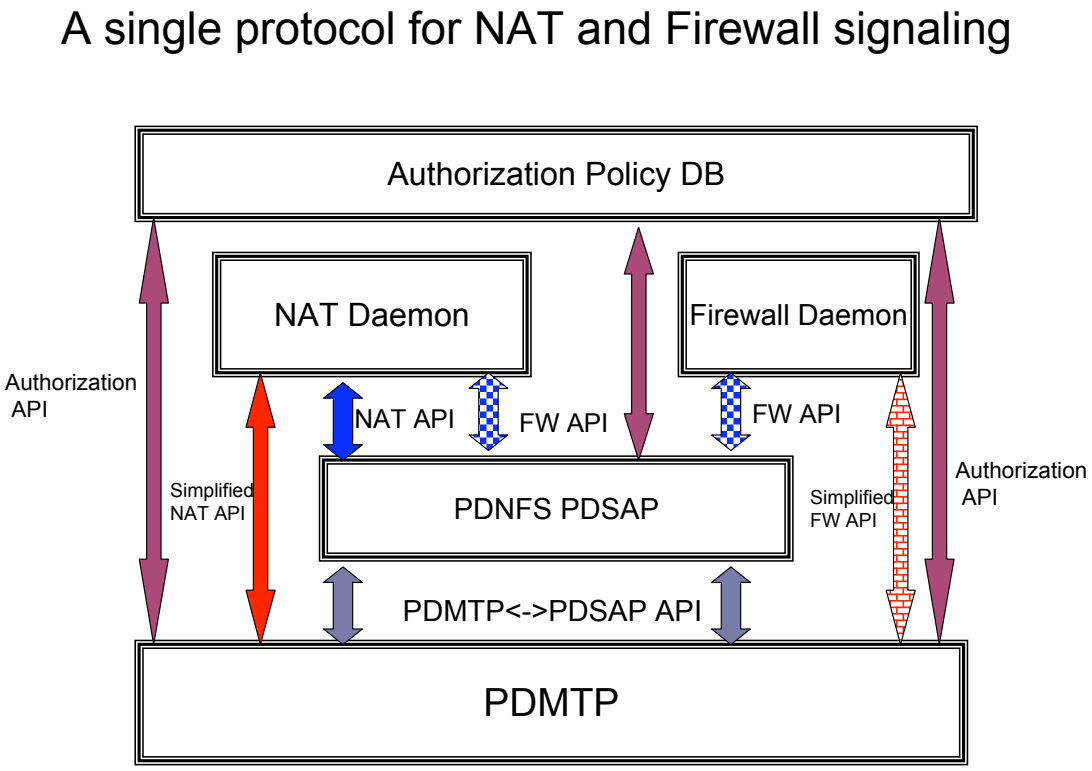


Figure 5.4: Interactions between all the PDSP layers and the NAT and Firewall daemons

5.3.6 Comparison Results

We have compared in simple deployment models both the split and combined signaling approaches by looking at the differing protocol semantics, performance and protocol layer interactions. We believe that the combined protocol signaling framework is more suitable based on

the summary provided in Table 5.1. In the following sections we shall be analyzing in greater detail the requirements for the combined signaling approach when deployed in the three network scenarios discussed above and in more complicated ones needed to handle backward compatibility.

Table 5.1: Comparison of the Pros and Cons of the split and combined signaling frameworks

Signaling type	**Performance**	**Semantics**	**Error correlation**	**NAT implementations**
Split	More messages[1]	Simpler	Cross PDSAP correlation	specific NATs[2]
Combined	Less messages	Relatively more complicated	No cross PDSAP correlation	All

5.4 Requirements for Handling a PDNFS Aware Application Host Behind One or More PDNFS Aware NATs

We have briefly discussed in section 5.3 deployments where more than one PDNFS aware NAT would be deployed in a network. In essence the problems resulting from such deployments are an amplification (shown in Figure 5.6) of what is called the "Intra-realm" address selection problem in [78]. This Intra-realm address selection problem, shown in Figure 5.5 occurs when two end-hosts do not know that they are hosted in the same network and hence that they can reach each other by using their physical interface addresses, hence back-hauling or connectivity issues[3] might occur.

A solution to the Intra-realm address selection problem would consist of delivering to the PDNFS node (the one which initiated a NAT bind creation) the list of translated data flow descriptors. Hence every traversed PDNFS aware NAT aware will include in the PDNFS NAT bind creation response message the translated address and port. As a result the PDNFS NAT

[1] As discussed earlier, pinhole refresh messages could refresh NAT binds and hence in total less refresh messages could be used

[2] Only NATs without filtering capabilities. Considering the number of NAT implementations with specific external host addresses in their NAT bind table, a split signaling solution is not applicable for widespread deployments

[3] Certain NAT implementations prevent outbound packets destined to the NAT's external interface address from being forwarded to internal addresses

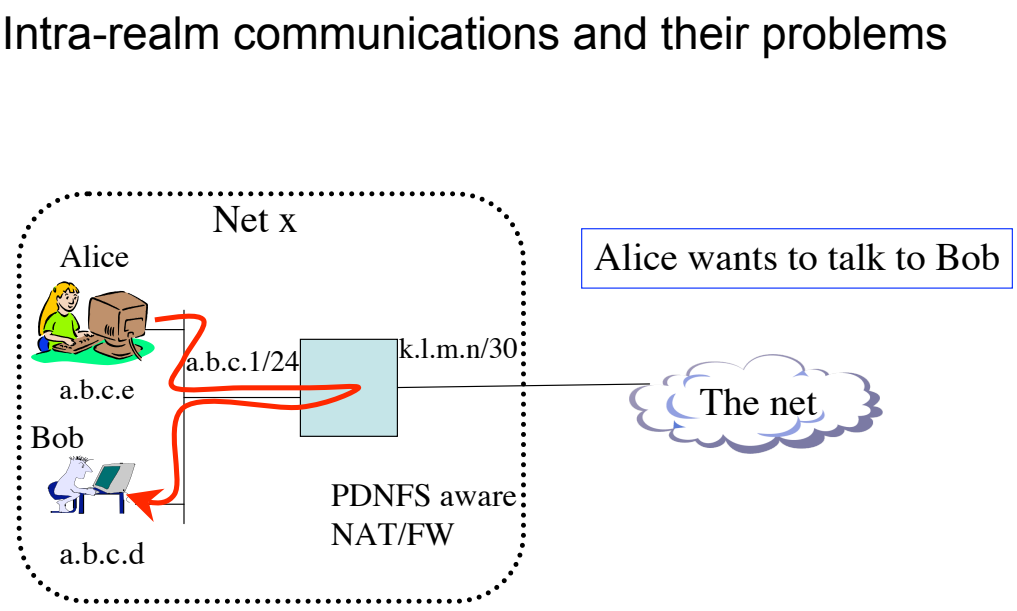


Figure 5.5: Intra-realm address selection impacts

layer would inform the user level application protocol that data flow packets could be sent not only to the host's physical interface address but also to a list of translated addresses (or mapped addresses).

Upon reception of the recipient node's addresses, the remote host would then send PDNFS messages to all the recipient node's addresses to allow its packets to be forwarded to the recipient node.

A specially designed NAT count object would be used in the message and incremented every time these PDNFS messages traverse a PDNFS aware NAT (which does not co-host a firewall function), and the final value of this counter when it reaches the SR will be returned in the response message. The remote end-host would then determine (based on the smallest NAT counter value) which recipient node address to use for sending PDNFS messages (and data flow packets).

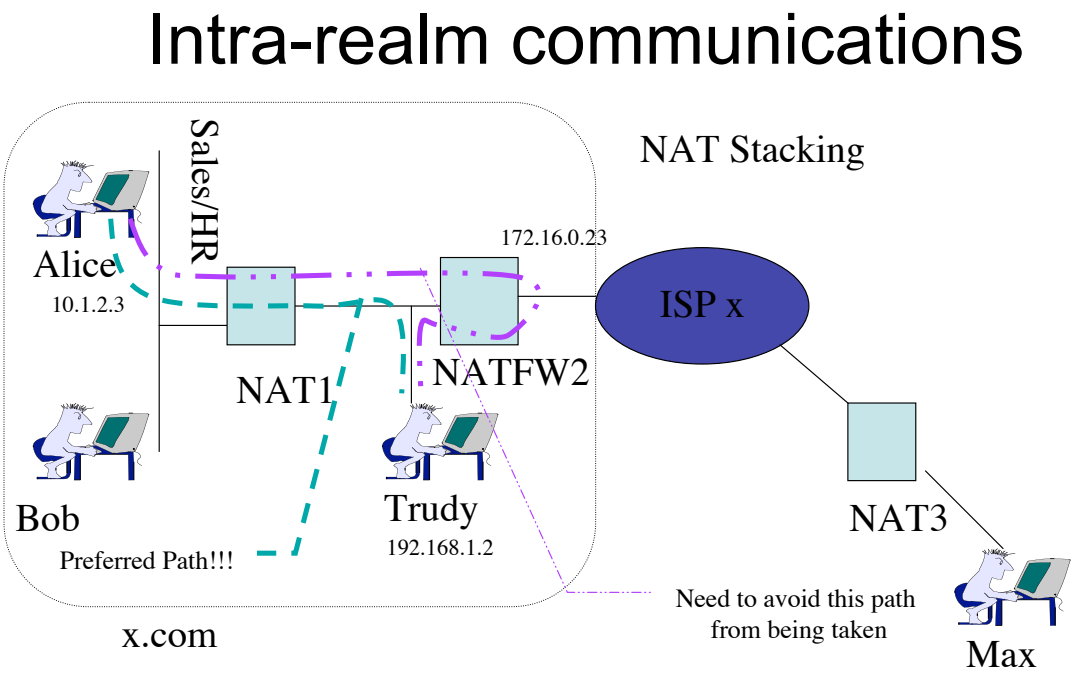


Figure 5.6: Impact of NAT Cascades on Intra-realm Address Selection

This solution might be subject to data flow hijacking (Figure 5.7) since a PDNFS host within the same network might have the same private address for its physical interface as the recipient node. We propose to use a nonce for verifying the source of the response message, that nonce would have been provided to the PDNFS SR at the user level application protocol layer (SIP or other).

5.4.1 Appropriate Security Model

In the previous chapter we have discussed the properties of PDMTP Messaging Associations which would have a one PDMTP hop scope. When multiple NATs are deployed within a network the NAT bind creation messages are forwarded within the same administrative network up to the edge-NAT. In certain networks authenticating a PDNFS host at the first PDMTP node might not be sufficient. In Figure 5.8 we show the difference between deployments where

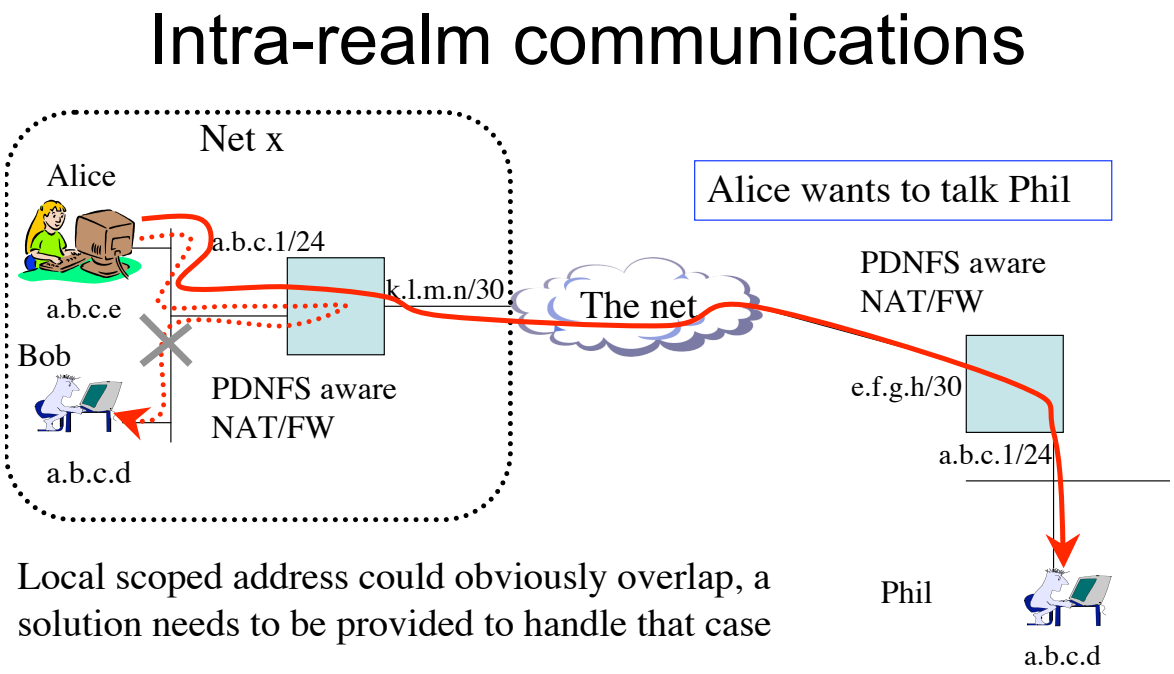


Figure 5.7: Data flow hijacking issues with overlapped addresses

a hop-by-hop authentication and authorization model in comparison with a model where the PDNFS SI/SR needs to be authenticated and authorized at every PDNFS node.

As opposed to the hop-by-hop security model (completely relying on the PDMTP layer for the authentication part), the end-to-middle security model will impact the PDNFS protocol semantics (this will be covered in section 5.9). Necessary semantic changes are required for the PDNFS protocol to allow the integrity protection as well as origin authentication of specific parts of the protocol messages, the entire message's integrity can't be protected as NATs might need to change the message's contents (this would depend on the outcome of the semantics analysis that we shall perform in section 5.9).

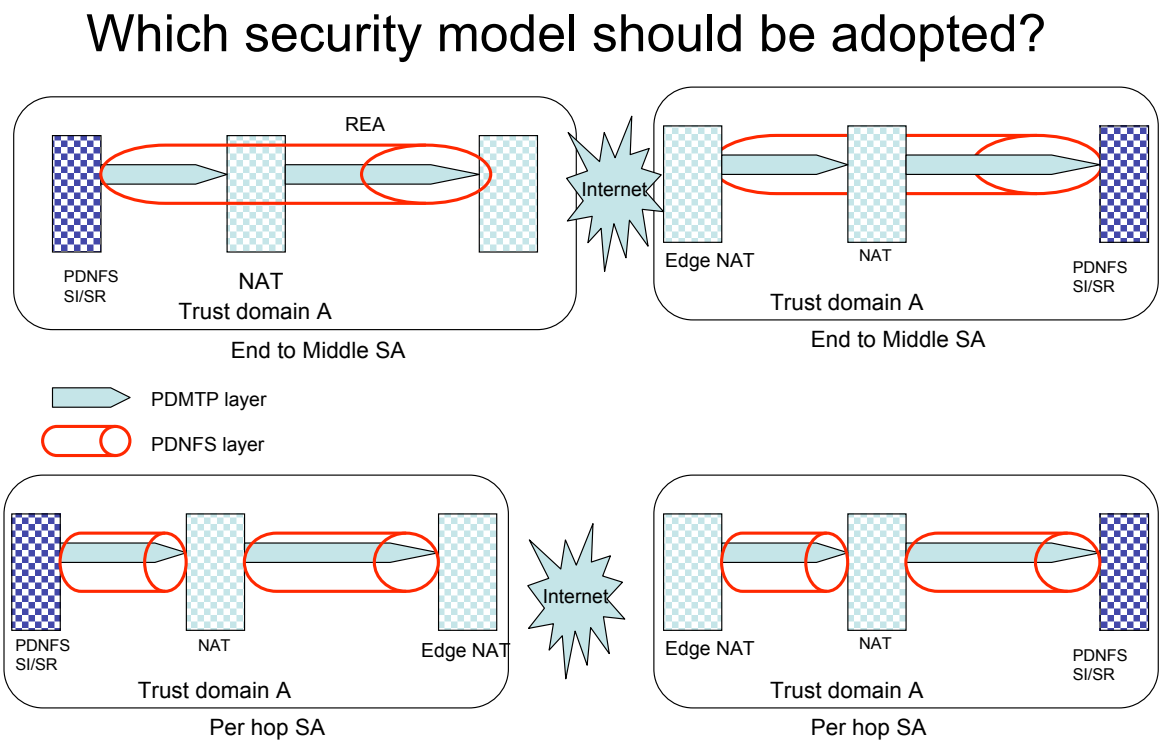


Figure 5.8: Comparison of Hop-by-Hop versus End-to-Middle security for NAT Cascades

5.5 Requirements For Handling PDNFS Aware Application Hosts Behind One or More PDNFS Aware Firewalls

From a payload handling perspective, traversing a cascade of Firewalls is simpler than a cascade of NATs.

However the operating procedures for signaling Firewalls are different to those for signaling a NAT[4] daemon on a networked system:

- The signaling message has to be forwarded from the PDNFS Initiator to its intended destination address (unless proxying and message path termination is required by the local network administrator)

- Since the message traverses both the PDNFS Initiator's network and the intended des-

[4]NATs without packet filtering capabilities

tination's (in general it is a PDNFS Responder co-hosted with a user level application client) network, the PDNFS Initiator request should be authenticated and authorized by all intercepting PDNFS nodes, inside and outside the PDNFS Initiator's administrative network.

Several methods could be used for this purpose:

- Hop-by-hop authentication by leveraging the PDMTP layer's authentication mechanisms and adding authorization mechanisms specific to the services requested by the PDNFS messages. This model is as weak as the weakest security measure to protect a PDNFS node from being subverted. When Firewalls deployed in different networks are peers for a specific PDNFS message, they cannot trust each other (unless each network establishes trust relations with each network[5] or with a specific set of networks as is done today with extranets). Hence this model is only suitable for use within the same administrative network or with "partner networks" or extranets.
- Direct authentication and authorization by the PDNFS entity receiving the message (as discussed in section 3.3.5). This approach does not scale well[6].
- Usage of a third party trusted by the PDNFS forwarder using any kind of assertion scheme (authorization tokens as discussed in section 3.3.5 or other)
- Usage of the PDNFS Responder (as a specific trusted third party): this method is suitable to use for Firewalls within the same administrative domain as the PDNFS responder. As opposed to the generic trusted third party approach this method does not require any specific changes to the network security infrastructure and would be favored in most deployments.

Assertion by the PDNFS Responder implies that authentication and authorization mechanisms exist to allow the PDNFS Responder to assert the validity of a PDNFS Initiator in its requests. When the responder to the PDNFS Initiator's request is a PDNFS Responder (and not a

[5]Since this model does not use any trusted intermediaries its complexity is in the order of $o(m^2)$ with m being the number of networks in the Internet

[6]Since this model does not use any trusted intermediaries its complexity is in the order of $o(n^2)$ with n being the number of PDNFS nodes deployed in the network

PDNFS SF acting as a proxy as discussed in section 5.9.4) the PDNFS Responder co-hosts a user level application client (for example a SIP UA); in that case the user level application client would have already instantiated a PDNFS responder and provided certain information that would authenticate the PDNFS Initiator (we have previously mentioned that this information could be in the form of a nonce). Hence the PDNFS Responder can assert that the PDNFS Initiator is requesting a pinhole opening that has been recognized by the trusted (from the perspective of the PDNFS forwarders) application end-host as useful. This assertion model puts specific constraints on the implementation of the PDNFS protocol on the applications' end-hosts as we shall see in Chapter 7.

5.6 Requirements for Handling PDNFS Aware Application Hosts Behind a Mix of One or More PDNFS Aware Firewalls and NATs

In hybrid deployments where a mix of PDNFS aware NATs and Firewalls are on the path of PDNFS messages, the requirements are a mix of the ones discussed in sections 5.4 and 5.5. We can distinguish the following subtle details in these deployments:

- PDNFS messages used to create NAT binds are intercepted by the Firewall but only handled at the PDMTP level to allow message forwarding; no PDMTP or PDNFS adjacency is established (as discussed in section 4.7)

- PDNFS NAT traversal: embedded addressing information specific to pinhole configuration in the PDMTP payload is translated by the PDMTP layer as defined in section 4.6. If additional addressing information was required, a specific RAO should be used to indicate that the message should be handled at the PDNFS layer for modifying the PDNFS message (and its implication in terms of MRS and MA use). Moreover since several NAT implementations include basic packet filtering characteristics, PDNFS messages should always be processed by the PDNFS layer (unless the NAT implementation is a pure address or address/port translator)

5.7 Backward Compatibility Considerations

Since the PDNFS protocol has not yet been deployed, backward compatibility with existing NAT and Firewall traversal is critical for the deployment of PDNFS.
We have already discussed in section 5.2 the various deployment models to be supported by the PDNFS protocol. In this section we shall only cover compatibility and coexistence with existing NAT and Firewall traversal mechanisms.

When the PDNFS protocol is not supported on an end-host, upon receiving PDNFS messages (assuming that the PDNFS message was forwarded to it and not dropped) the receiving end-host will simply send an ICMP error message with port unreachable. Since the protocol supports deployments where only one end-host supports the protocol (this is in reference to the incremental deployment requirement that we have discussed in sections 5.2 and 3.2, compatibility with the existing NAT and Firewall traversal solutions used by remote end-hosts should be easily achieved.
In Chapter 2 we have described the various existing NAT and Firewall traversal schemes for which the following compatibility analysis is applicable:

- Virtual Private Network (VPN) solutions: by their nature VPN solutions by-pass NATs and Firewalls, and hence there are no interactions with the PDNFS aware Middleboxes. When two PDNFS aware end-hosts would try to communicate with each other across a VPN, they would send PDNFS messages to each other across the VPN, and no PDNFS aware Middleboxes will be traversed.

- STUN [9]: handles only NAT traversal[7] which it does by providing the translated address the application client.

- Media Proxy: handles only NAT traversal[8], and no incompatibility issues are foreseen. The application end-host could be using both this method and the PDNFS PDSAP and would be able to decide which method to use after receiving responses to its translated

address determination request.

- Application Layer Gateways: no incompatibility issues are anticipated. The application end-host at one end of the communication (without PDNFS support) could be deployed within a network using ALGs.

- Targeted Middlebox signaling protocols (MIDCOM, RSIP and UPNP)

We shall analyze coexistence requirements for the PDNFS protocol with all the above NAT and Firewall traversal approaches.

5.7.1 Compatibility with STUN Deployments

In most cases STUN clients would have already been installed on application end-hosts prior to the installation of the PDNFS SI/SR agent. On an application end-host priority should be provided to the PDNFS PDSAP to detect if there is a NAT by sending a NAT bind creation message related to an application session.
If the application was not authorized to access the PDSAP API (this will be discussed in Chapter 7) due to local security constraints, the STUN client would be used instead. Another reason for the application to prefer STUN as the NAT traversal solution would be if the PDNFS SI or SR has reported authorization error issues. If both STUN and PDNFS are available to an application and it is able to use either STUN or PDNFS depending on which is available, they should be signaled in parallel due to the real-time sensitivity of many such applications as there are no means to determine in advance the reactions of the networks Middleboxes. No incompatibility issues are anticipated with STUN deployments. One of the application end-hosts in communication could use STUN whereas the other could use the PDNFS PDSAP solution.

[7]Firewall traversal could be handled with specific engineering rules by using a stateful UDP pinhole for outbound traffic but without any external node address restrictions

[8]Firewall traversal could be handled with specific engineering rules by using a stateful UDP pinhole for outbound traffic with optional restrictions set to the well known Media Proxy addresses

5.7.2 Compatibility with Targeted Middlebox Signaling Solutions

There are no constraints to implementing a PDNFS entity (SI/SR/SF) on the same node as a MIDCOM, RSIP or UPNP functional entity. Within an application session one of the end-hosts in communication could be using one of the known targeted signaling solutions (if applicable[9]) and the other could use the PDNFS protocol.

5.8 Avoiding Non-Deterministic Behaviors

Since we are proposing to use a single protocol for signaling NAT binds and Firewall pinholes, care must be taken to avoid getting into states (especially when NAT and Firewall daemons are co-hosted) where the PDNFS implementation on a node is unable to readily determine how it should react to a message or which type of message it needs to send when a given event occurs. Since the ability to initiate and request specific resources from Middleboxes is not tied to the node's role in the application data exchange, it is possible that certain PDNFS instances that might have anticipated that they could only be Signaling Initiators might at some point become Signaling Responders:

- A node behind the NAT will initiate a PDNFS signaling exchange to create a NAT bind for the forwarding of inbound data flow packets, however that node would also initiate PDNFS signaling exchanges for the forwarding of its outbound data flow packets. The remote end-host, if it supports the PDNFS protocol, would initiate PDNFS signaling exchanges to allow forwarding of its packets (by sending a PDNFS message to the local host's externally reachable address). In the scenario described, a PDNFS instance starts as Initiator (for NAT bind creation purposes) of a signaling exchange and then changes to become both Responder (for the downstream path) and Initiator (for the upstream path).

- For backward compatibility support, a PDNFS entity should be able to cope with deployments where no Signaling Responder responds to the PDNFS messages. The last

[9]RSIP allows only the creation and maintenance of NAT binds, UPNP's scope is limited to one LAN segment due to the use of Link Local Multicast Naming Resolution

PDNFS node (a PDNFS aware NAT or Firewall) on the downstream path should be able to fulfill the role accomplished by the PDNFS SR in returning PDNFS messages. That last PDNFS aware NAT and Firewall on the path must have a deterministic behavior for every received message.

- Moreover backward compatibility mandates that an application aware end-host be able to use the PDNFS protocol without any knowledge of the PDNFS support on the remote application end-host. This implies that by default the PDNFS aware application end-host would act as an Initiator for both its inbound and outbound data packets (this would include PDNFS aware end-hosts behind PDNFS aware NATs and Firewall)

We can summarize this discussion by acknowledging that the use of a signaling message direction (and its implicit relation to the characterization of a node by being a Signaling Initiator or Signaling Responder) at the PDNFS layer (and in general for any PDSAP) adds an unnecessary ambiguity in the protocol framework. Hence the concept of Signaling Initiator and Responder is still applicable from a message routing perspective but PDNFS (and other PDSAPs) instances should be characterized based on their role on the packet flow's direction. Therefore we shall define the following characterization for PDNFS nodes:

- PDNFS Inbound (**PDI**): PDNFS Inbound data flow agent, uses PDNFS signaling to allow the forwarding of inbound data flow packets
- PDNFS Outbound (**PDO**): PDNFS Outbound data flow agent, uses PDNFS signaling to allow the forwarding of outbound data flow packets

A PDNFS SF remains as a PDNFS node (that co-hosts Middlebox functions) that forwards and acts upon PDNFS messages. This entity could also potentially need to behave as a PDI or PDO.

5.9 Detailed Protocol Operations

In the previous sections we have gathered all the requirements to determine the proper semantics for the PDNFS protocol. In this section we shall describe how the protocol operates and what

types of information should be exchanged between the PDNFS entities to meet the protocol framework requirements. Any PDNFS request of any type, be it pinhole or binding creation, modification, refresh or removal has to be authenticated and authorized. The appropriate mechanisms for authentication and authorization will be discussed in section 5.13.

5.9.1 NAT Bind Creation and Mapped Address(es) Determination

Every application known to have NAT traversal issues (as described in section 2.4.2) will request the PDNFS layer to instantiate a PDNFS Inbound agent. The application request will provide sufficient information to request the creation of one (or several if there are many NATs in the network) NAT bind(s) with the associated mapped addresses and ports.

The PDI instance sends a specific message, that we shall call the Mapped Address Determination (**MAD**) message , with appropriate instructions to the PDMTP layer (a specific RAO value is set to avoid PDNFS handling on PDNFS aware Firewalls[11]). The message is sent to an arbitrary address, the Opportunistic Address (OA) set to ensure that the message is forwarded towards the edge of the addressing realm to reach the (or one of the many if applicable in case the network is multi-homed) edge-NAT of the administrative domain. The MAD message is given to the PDMTP layer via the PDMTP API with message reliability, integrity protection and confidentiality as Message Transfer Requirements (MTR) as defined in Chapter 4. Intermediate PDNFS aware Firewalls[12] will intercept messages for PDMTP forwarding purposes but will not analyze MAD messages due to the hints provided by the RAO setting. Intermediary PDNFS aware NATs (NAT1 and NAT2 in Figure 5.9) intercept the message at the PDNFS level and forward the message downstream towards the OA. When forwarding the message these PDNFS nodes will reserve a NAT bind (and hence allocate a mapped address and port pair). However the NAT bind will not be enabled until a positive response message is received from a downstream edge-NAT.

Upon reaching the edge-NAT, the PDNFS MAD message will be intercepted and will induce a PDNFS Response (**PDRESP** defined in section 5.14.3) message to be sent back towards the PDI. The PDRESP message will provide a mapped address and port pair(s) allocated on each

[11]This obviously does not include Firewalls co-hosting a NAT daemon

[12]PDNFS Edge-Firewalls will behave differently as we shall see in subsection 5.9.4

traversed PDNFS aware NAT.

Figure 5.9 summarizes the usage of the MAD message and its associated response, the PDI and PDO agents are shown in the figure but the interactions with the MAD messages and responses are for the PDI agent. Specific MAD requests and options should be available such

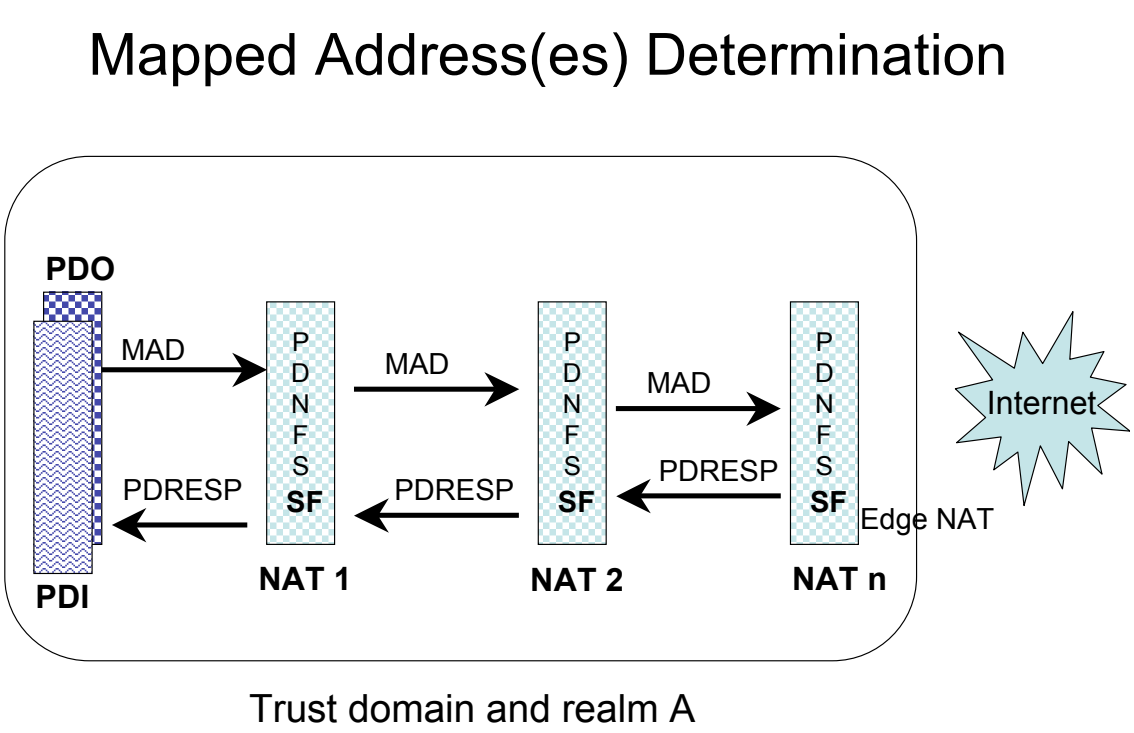


Figure 5.9: Mapped Address Determination Message Use

as:

- Ability to request one of the following: a preferred mapped address, a preferred mapped port or a preferred mapped address and port pair.
- Port parity preservation (i.e., whether an even or odd numbered port should be allocated).
- Contiguous port allocation with port parity preservation (this would be used by applications having RTP/RTCP data streams and not implementing [77]'s recommendations)

- Ability to request a specific lifetime for the NAT bind
- Nonce: As discussed in section 4.6, when creating NAT bind entries it is important to use nonces to avoid message flooding attacks by rogue PDMTP SIs. In cases where the nonce is not in the form of a hash of a unique end-host's local scoped address, the MAD message should include a nonce[13] that would be part of the NAT bind entry and used when receiving PDISC PDMTP messages and PDNFS messages.

Since one of the protocol's property is to successfully operate in unilateral support scenarios (which would typically happen when the protocol is initially deployed in the Internet), messages initiated by the PDI instance will be used for requesting or triggering the creation of pinholes for inbound flows.
The MAD message is resent periodically until an inbound pinhole, creation or refresh message is received by an authorized PDO. The usage of the MAD message for this purpose is described in section 5.9.4.

5.9.1.1 Handling Edge-NAT Configuration Issues

Configuration errors can often prevent protocols from being properly deployed and used. In the particular case of the mapped address determination process, a system administrator could forget to configure a PDNFS aware NAT on the edge of the addressing realm as an Edge-NAT. Since the mapped address determination process relies on a response from the Edge-NAT, this type of error configuration could have a dramatic impact on the protocol.

We propose to solve this issue by using the PDMTP's error handling procedures when no response is received for a PDISC message (in the case that no issues arise with the address spoofing, the PSI would receive an ICMP port unreachable error, or else no response would be received). In case the PDMTP layer does not succeed in transmitting the PDISC message it informs the PDNFS layer (via the PDMTP$<->$PDNFS API) about the transmission error, hence the intermediary NAT will behave as an Edge-NAT.

[13]Since the nonce is sent to the remote application end-host, the nonce type will impact the semantics of the application protocol. The current default nonce based on the local end-host's interface local scoped address is already available for SDP based protocols using [74]

5.9.1.2 Semantics for MAD Related Errors

The protocol should allow the reporting of the following errors transmitted in response to unsuccessful MAD message processing:

- PDI authorization issues: unauthorized to request for a NAT bind
- Mapped address pool is exhausted
- Mapped port pool is exhausted
- Requested mapped address and port pair is already allocated
- Unavailable resources: memory or processing power
- System issues: host under maintenance, kernel panics etc ...

Certain error types might be associated to a period of time after which the PDI should retry its request. Whether this applies depends on the precise cause of the error condition.

5.9.1.3 Opportunistic Address (OA) Selection

A PDI agent instance sends signaling messages prior to the reception of a data flow source address, hence a different address has to be selected when sending PDI messages. The address selected for that purpose will be known as the Opportunistic Address.
This address could be either a 'well-known address' utilized by any PDI agent, an administratively defined address appropriate for any PDI agent in the domain (i.e., appropriate given administrative domain's address allocations and routing tables) or a host/application specific address (i.e., specific to a particular host or application).
There are two main characteristics for this address:

- Packets sent to this address should all be routed towards the edge of the administrative's domain addressing realm and trust domain (the same address could also be used for enabling packet forwarding on Firewalls as discussed in section 5.9.4)
- PDNFS unaware Middleboxes at the edge of the addressing realm or trust domain should be able to prevent the forwarding of messages to that address. Failure to prevent the

message forwarding could create attacks on the nodes to which the OA is assigned if such nodes exist.

We propose to use a well-known address to meet the above requirements in all network deployments, however implementors of this technology could use any address that would meet these requirements within their own deployment scenarios.

5.9.2 State Indexing and Correlation Between PDI and PDO Initiated Messages

In Chapter 4, we have discussed the use of a data flow's destination address and port (in an inbound PDMTP message on external interface as defined in Figure 2.5) and a nonce in redirecting a PDMTP message to its appropriate destination.
At the PDNFS layer, correlation information is not only used for message forwarding purposes but also for overall protocol framework performance optimizations. As discussed in section 5.3.6, the combined protocol approach could have the advantage of reusing pinhole refresh messages to refresh NAT binds. This statement could only be true if messages sent by the PDI, to create and refresh NAT binds, are correlated to messages sent from a PDO to the previous PDI to create and refresh pinholes. In addition the ability to correlate the PDO initiated messages to the PDI initiated messages will allow the framework to remove PSI and PSR instances (and their associated MRS entries) associated to the MAD (or other PSI initiated) message routing state (which would no longer be required after mapping the cross-reference).

We shall use the following conventions to index the state and correlate it to related PDI and PDO messages:

- Upon reception of a PDI message, a PDNFS SF on a PDNFS aware NAT, instantiates a PDNFS SF agent that will be bound to the data flow and proposed nonce (inherently binding it to the NAT bind created in the NAT daemon's NAT bind table). This PDNFS instance will be indexed by using a combination of the mapped address and the associated nonce.

- The MRS entry at the PDMTP layer is inherently bound to the PDNFS instance created above, for which one PSI and one PSR instance have been created.

In Figure 5.10 we show the various instantiations of PDMTP and PDNFS agents required on one NAT, the same is applicable to NAT2 and NAT3 in the example shown.

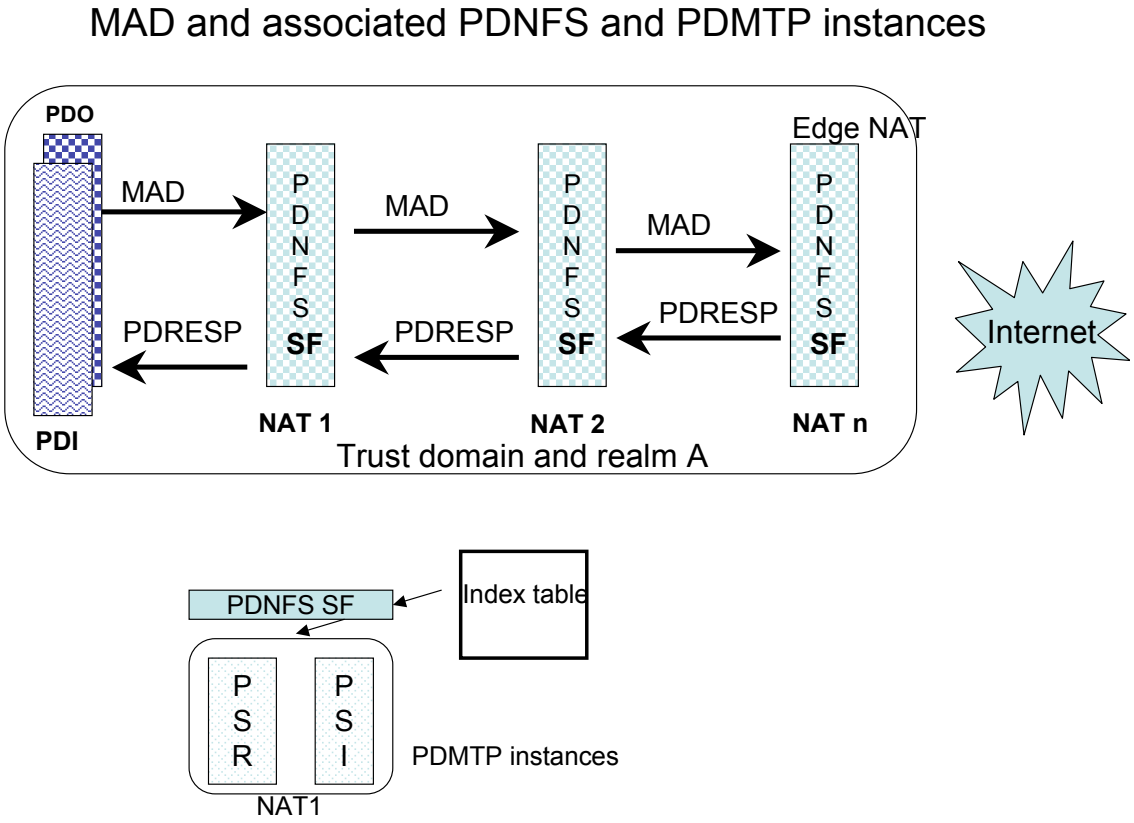


Figure 5.10: PDNFS SF instances and correlating PDO and PDI initiated messages

Upon reception of a PDO initiated message, the PDNFS managing agent analyses an indexed table in which indexes are based on the mapped addresses and associated nonces. Once an entry is found the received PDO initiated message will be bound to that indexed entry and no new PDNFS SF instance would be created. Upon successful response (by the PDI instance which initiated the creation of the PDNFS instance) to the PDO initiated message, there would be no need for the PDI to continue refreshing the NAT bind and the MRS entries used for routing the previous MAD messages would no longer be required.

5.9.3 Enabling Outbound Data Flow Packet Forwarding: Operations and Semantics

A user application known to have Firewall issues (based on section 2.4.1) will instantiate a PDO instance and provide the PDO with the appropriate information to open a pinhole on Firewalls traversed by the data flow packets. In several user level applications, the data flow receiver's address and port might not be known when initiating the application's session (we will use SIP [4] in the following example), as shown in Figure 5.11 some delay is required prior to sending the PDNFS message towards the message target. Once the message target is known, the PDNFS message is sent by using the standard PDMTP API calls (discussed in Chapter 7) with a specific requested RAO value, indicating that message as a PDNFS pinhole configuration message. The PDISC message indicates the need for an MA with message reliability, integrity protection and confidentiality as Message Transmission Requirements. The

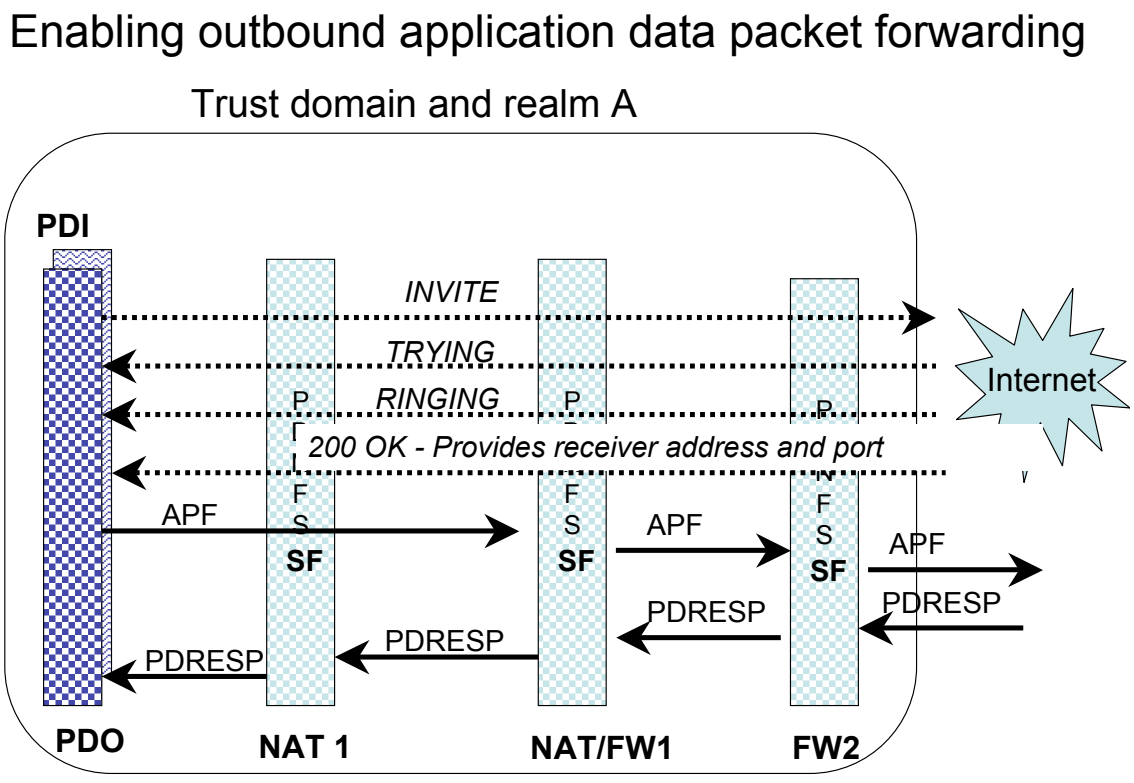


Figure 5.11: Outbound data flow pinhole creation and maintenance

Allow Packet Forwarding (**APF**) message shown in Figure 5.11 could be used not only for pinhole creation but also for maintenance of the pinhole[14].

Traversed PDNFS aware NATs (without packet filtering support) behave differently with APF messages (than PDNFS aware Firewalls).

- Based on the results of section 5.9.2, inbound (received on the NAT's external link as defined in Figure 2.5) APF messages are handled at:
  - The PDMTP level: nonce validation, redirection to the PDI instance host and data flow descriptor translation.
  - At the PDNFS level, the NAT count provided by the PDMTP layer is incremented (only if the message was embedded in a PDMTP message without an existing MR-SUID in the MRS table on the intercepting node[15]) and the procedures discussed in section 5.9.2 is applied. Therefore no new PDNFS SF instance is created and upon receiving a successful response from the PDI instance (which created the PDNFS SF instance and its associated NAT bind), MAD messages would no longer need to be sent periodically by the PDI instance. Moreover, the PDNFS SF instance will be maintained by periodically transmitted APF messages.
- For outbound (received on the NAT's internal link as defined in Figure 2.5) APF messages, messages are handled at the PDMTP level as discussed in section 4.6: data flow descriptor translation and notification to the PDMTP SI is carried out as discussed in section 4.6.

Every PDNFS aware Firewall traversed by a PDMTP PDISC message with the APF PDNFS specifically assigned RAO value intercepts the PDMTP PDISC message and establishes a secured and reliable Messaging Association with its upstream peer. The PDNFS APF message is then processed, the pinhole is created and the message is forwarded (with the same initial MTR provided to the PDMTP layer) downstream towards the message's target address. In certain networks message forwarding might not be accomplished by edge-Firewalls due to local network policies, we shall discuss these cases in Chapter 6.

[14] in maintenance we include pinhole refresh, modification and removal

[15] This is required to avoid incrementing the NAT counter several times during intra-realm communication cases as discussed in section 5.9.3.1

The APF message semantics will be discussed in section 5.14.2 and tailored to handle the operations discussed above.

5.9.3.1 Usage of the APF Message for Solving Intra-Realm Issues

As previously discussed in section 5.4, a solution to solve intra-realm issues would be to send a PDNFS message from one application end-host to the other while incrementing a NAT counter at each NAT traversal (at the PDMTP level or the PDNFS level depending on the message direction with regards to the network in which the PDNFS aware NAT is deployed). The APF message is sent from one end-host to the other but it is handled at the PDMTP layer for outbound messages on PDNFS aware NATs and at both the PDMTP and PDNFS layer for inbound message on PDNFS aware NATs. The PDMTP's PDISC NAT counters are used as specified in Chapter 4; every PDNFS aware Firewall (or Firewalls co-hosting a NAT capability) would receive the NAT count and update a NAT counter embedded in the PDNFS APF message. A NAT and Firewall will increment the NAT counter received at the PDMTP layer as it will be handling the PDNFS APF message.

Upon reception of a list of addresses where an application end-host could receive data flow packets, the application triggers the creation of multiple PDO instances[16] (one for each received address), a PDO management task monitors the responses from the various PDO instances and decides to keep the PDO instance that has received the lowest NAT counter value.

In Figure 5.12, two PDOs were instantiated where the first APF$<->$PDRESP exchange results in a NAT counter value of 1 and the second APF$<->$PDRESP exchange (using interface addresses and not a mapped address as used with the first exchange) results in a NAT counter value of 0 and is selected.

5.9.3.2 APF Message Specific Errors

The protocol should allow the reporting of the following errors transmitted in response to unsuccessful APF message processing:

- PDO unauthorized to request a pinhole for a specific component of the data flow descriptor

[16]This implementation detail will be discussed in Chapter 7

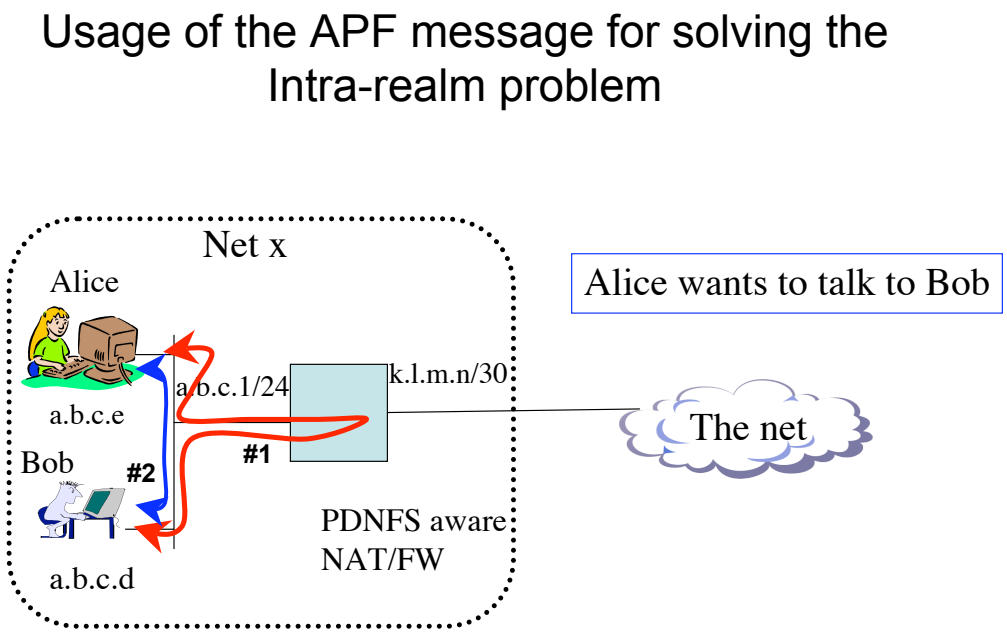


Figure 5.12: Use of the APF message to Solve the Intra-realm Problem

- Unavailable resources: memory or processing power
- System issues: host under maintenance, kernel panics etc ...
- Requested pinhole lifetime is unacceptable

Certain error types might be associated to a period of time after which the PDI should retry its request. Whether this applies depends on the precise cause of the error condition.

5.9.4 Protocol Operations in Unilateral Usage Scenarios

Unilateral operation of the PDNFS protocol is mandatory when one of the application end-hosts does not support the PDNFS protocol.

We can distinguish two types of unilateral signaling scenarios:

- Scenario 1: Responses to PDNFS signaling messages are made by the last PDNFS aware host (a Middlebox in most cases[17]) along the path rather than the receiving end-host that does not support the PDNFS protocol. This scenario occurs when enabling outbound data traffic for "send-only" unidirectional applications.

- Scenario 2: Responses to PDNFS signaling messages are made by the last PDNFS aware host (a Middlebox in most cases[17]) along the path, which in turn initiates signaling messages to the PDI indicated in the originally received message. This scenario is used by bi-directional applications as well as "receive-only" unidirectional applications.

Since an application might not always provide end-host capabilities (in terms of signaling protocols such as PDNFS or any other PDSAPs), an application end-host does not always know if the remote end-host in communication supports the PDNFS protocol. Hence a PDNFS aware end-host should always assume the worst case and operate in unilateral signaling mode.

5.9.4.1 Scenario 1 Mode Of Operation

Any application requiring outbound packet traversal instantiates a PDO instance, which in turn issues an APF message once the destination address is known together with the remainder of the data flow descriptor. This scenario shows the typical use of the APF message when the message's destination does not respond to the PDMTP PDISC message while establishing a MRS on the last downstream PDNFS aware Firewall or NAT and Firewall Middlebox. After retrying the maximum number of allowed PDISC transmissions, the PDMTP layer will notify the PDNFS layer (via the PDMTP API) that the message can't be delivered.
The PDNFS layer will then respond back positively (i.e., outbound packet forwarding has been enabled) to the PDO instance, while indicating that the PDI was not reachable. This successful PDRESP reflects a potentially optimistic view that there are no other firewalls downstream of the node sending the PDRESP. In case the PDI reachability problem was due to a transient network connectivity issue or data flow packets were not being forwarded properly, the application protocol[18] should have the ability to indicate to the PDO that packets were not received

[17]We shall discuss in Chapter 6 how a PDNFS aware system could have a PDNFS PDI/PDO role without either being application aware or a Middlebox

and hence another APF should be sent to the data flow receiver. If the issue persists after two APF transmissions, the PDO can conclude that the data flow packet forwarding towards the specific destination is not possible and notify the user application about the problem.
This behavior could either be triggered because of a real PDI reachability problem or due to local administrative network policies as we shall discuss in Chapter 6.

5.9.4.2 Scenario 2 Mode Of Operation

The application triggers the instantiation of a PDI agent which will send a MAD message to detect if there were any NATs in the network and at the same time that message will trigger the sending of an APF message by either an edge-NAT or an edge-Firewall.
There are some incompatibility issues between this proposal and what we have discussed in section 5.9.1:

- A MAD message is not forwarded by an edge-NAT
- A MAD message does not embed the address of the data sender (as it is not always available to the application when the message has to be sent), this implies that another MAD message would need to be sent to update the Edge-NAT or the Edge-Firewall about the data sender address and port
- A PDRESP response message to a MAD message normally embeds a mapped destination address and port pair (or several sets) as well as a mapped source address and port pair (for twice NATs). The PDRESP response message semantics should also consider cases where no NATs are found.
- Since the PDI is triggering the edge Middlebox (the one at the edge of the administrative domain that could be either an edge-Firewall or an edge-NAT), the PDI should be refreshing the PDNFS state on the edge Middlebox (as well as all the other intermediaries for the NAT bind states)
- PDNFS Firewalls do not handle MAD messages at the PDNFS layer, only PDMTP message forwarding is enforced without any Messaging Association establishment by the

[18]Streaming applications normally support such capability [79]

intercepting Firewall. The Edge-Firewall will have to behave differently.

We propose to extend the MAD message to incorporate the data sender address and port in its semantics as well as the ability to use the message for PDNFS state refresh (and associated Middlebox resources: NAT binds and Firewall pinholes), data flow descriptor updates and NAT bind removal (inherently triggering the Edge-Middlebox to send an APF message to remove PDNFS SF instances created by the APF as well as all associated middlebox resources). Moreover the MAD message's forwarding will not only be subject to the verification that the traversed NAT is at the edge of the administrative domain's addressing realm, but at the edge of the overall administrative domain; otherwise the message would be forwarded towards the OA. This also implies that a Firewall at the edge of its administrative network will need to be configured as an Edge-Firewall (this characterization was not necessary for the use of the APF message).
In Figure 5.13 we show the relations and the possibility of several trust domains and addressing realms co-existing within the same administrative network. In that example several trust models exist within the same administrative network, some networks could have several addressing realms as well.

To allow the ability to use the normal MAD message for applications having in-built capability negotiation mechanisms, an optional flag (that we shall call "Bilateral Mode") will indicate to Middleboxes when the unilateral signaling mode should not be used.
For "receive-only" type applications, a PDI instance is created and sends a MAD message with the above extensions. For these types of applications only a PDI instance would be instantiated whereas bi-directional applications would require the instantiation of a PDO as well and the APF message would be sent for the outbound packet flow as discussed in sections 5.9.3 and 5.9.4.1.
Every Firewall traversed by the MAD message will follow the rules discussed in section 5.9.1, an Edge-Firewall would be an exception as it would intercept the message.
Node behavior on receiving a MAD message is dependent on the type of the intercepting node:

- An intermediate PDNFS aware NAT will act as if the message was a MAD message as

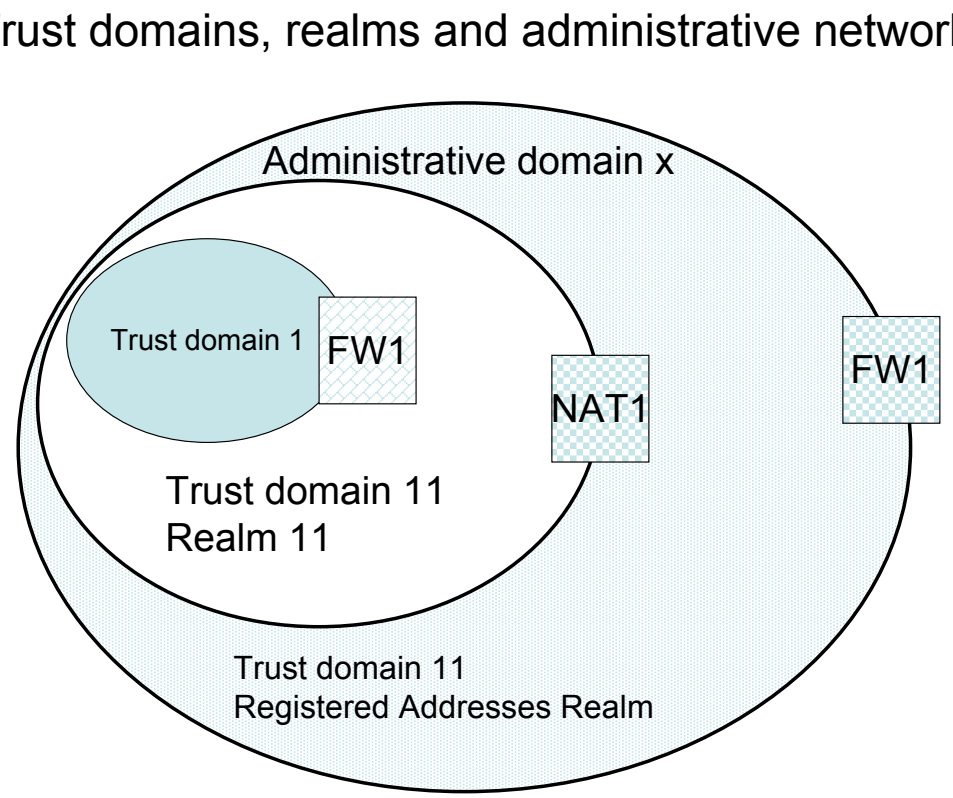


Figure 5.13: Trust domain and addressing realm delimitation

defined in section 5.9.1

- An Edge-NAT will behave as an intermediate NAT (if the bilateral behavior flag was not set for the message and if it was not the Edge-Middlebox), or else the behavior is the same as for an Edge-NAT in section 5.9.1; otherwise if the Edge-NAT is the Edge-Middlebox it will respond with a PDRESP message and then send an APF message to the PDI's address and port.

- An intermediate PDNFS aware Firewall will not handle the MAD message at the PDNFS layer (as done in section 5.9.1).

- An Edge-Firewall will behave as an intermediate PDNFS aware Firewall if it was not the Edge-Middlebox, else it will respond back with a PDRESP message with the original data flow descriptor and mapped addresses corresponding to the original data flow descriptor.

After transmitting the PDRESP message the Edge-Firewall sends an APF message to the PDI's address and port.

MAD messages will be repeatedly (until the user application session no longer need inbound data flow packets) transmitted until an APF message is received from a PDO instance. In the event that APF messages were not received from a PDO instance and the PDI instance has stopped sending MAD messages, all NAT binds will expire as well as inbound packet flow pinholes.

A mechanism should be used to allow the PDI to distinguish APF messages sent by the Edge-Middlebox from the ones sent by a PDO instance, there are several solutions that could be used:

- The first received APF message is always assumed to be sent by the Edge-Middlebox (when the MAD "Bilateral Mode" was not set). This solution could potentially result in a race condition where APF messages sent by the PDO could arrive first.
- The MAD message's response includes the MRSUID which will be used for routing the Edge-Middelboxes APF message, or any other information that would allow the PDI to identify APF messages sent by the Edge-Middlebox
- The APF message will incorporate a flag that would indicate the type of node which generated the message

We prefer the second solution as it would require less configuration on Edge-Middleboxes to verify if the flag is properly set. We shall consider that solution while defining the semantics of the PDRESP message in section 5.14.3.

There are no differences in MTR requirements between MAD messages used in either mode ("Bilateral mode" and default mode), and the RAO setting remains the same as the one as discussed in section 5.9.1.

5.9.5 Asynchronous Notification Semantics and Usage

In the event that failures occur which might affect the data flow's packet forwarding[20], the PDNFS aware Middlebox detecting such a problem should notify the PDNFS entity which has

initiated the packet forwarding[21]).

Although providing such feedback is interesting to the PDNFS entities which have triggered the enabling of the data flow packet forwarding, this ability is redundant with some applications' feedback mechanisms about the data flow packets. In section 3.4 we have discussed such applications' inherent feedback mechanisms or their use of transport protocols with such mechanisms. Hence the protocol should allow a PDI or PDO, when requesting the sending of a MAD or APF, to also request the sending of notification messages in case the associated data flow packets' forwarding could be impacted by a failure that could be detected along the path. The default usage should be to use the protocol without asynchronous notifications as for most NAT and Firewall impacted applications, the data flow has failure detection mechanisms (e.g., RTCP receiver reports, the application control plane sends notification messages or TCP or SCTP acknowledgments are not received).

If used, asynchronous notifications should only be sent by PDNFS SFs or the PDI (if the packet forwarding was enabled by the PDO) which are handling the affected data flow. A simple check could be made by the PDI or PDO to validate if the notification message was sent by a valid PDNFS instance: the notification message would have been sent on an already established MA associated with the MRS entry used by previous PDFNS messages and indexed with the MRSUID.

When asynchronous notifications are used, avalanche conditions could occur in the event of major outages in the network and notification messages could flood nodes hosting the various PDI or PDO instances. Since the problem is related to the handling of several PDNFS messages, message aggregation could be used but would break the assumptions that we have previously discussed:

> The reception over an existing MA associated with the MRSUID indexed MRS entry, is used as a simple mechanism to validate a notification message sender. Performing message aggregation for notification messages related to several SBSs, which might have

[20]for outbound or inbound direction when considering the administrative network in which the application end-host resides

[21]a PDO when bilateral and unilateral signaling mode is used (for "send-only" applications) or the PDI when unilateral signaling is used

been instantiated by using different MAs might not be effective.

Therefore an alternative solution would be required to allow a PDI or PDO to validate a notification message source.

We propose to solve this issue by using some alternative cross-reference information that would not be bound to the way in which the message was routed (which was the case of the MA). Such information is already exchanged in PDNFS messages creating Middlebox resources: nonces used to allow message forwarding and cross-referencing a PDNFS SF state instantiated by a PDI message to a PDO initiated message. Any selected cross-referencing material should be anti-replay proof and be already linked to an impacted data flow.
The semantics of the optional asynchronous notification message, **NOTIFY**, for the PDNFS protocol will be discussed in section 5.14.4. The main semantic requirements would be to embed the same error object types as the ones defined for the PDRESP response message, except that the temporal effect attribute of the error would not be applicable as the notified errors are affecting on-going packet flows.

5.10 PDNFS Instance and Middlebox Resource Refreshing Rules

PDNFS instances are created on application aware end-systems as well as on Middleboxes and standard IP routers when the Path-Directed Trigger Signaling (PDTS) PDSAP discussed in section 6.1.3 is used. On an application aware end-system either a PDI or a PDO or both could be instantiated for the duration of an application session. These instances are bound to the lifetime of the application session.
An IP router triggered by the PDTS PDSAP could instantiate both a PDI and PDO, the PDI and PDO instances would then be bound to the lifetime of the PDTS PDSAP Responder which would have to be refreshed periodically based on the application session's duration (discussed further in section 6.1.3).

PDNFS SF instances are created on Middleboxes and are refreshed by PDI initiated messages when unilateral signaling is used (MAD sent by the PDI and APF messages are sent by the Edge-Middlebox) or by PDO messages when bilateral signaling is used (APF messages are

sent).

A PDNFS SF instance has to fate share with its associated Middlebox resource handling the data flow packet traversal. In case the PDNDS SF instance is not refreshed the Middlebox resources should be removed. Moreover, in case an explicit PDNFS SF instance removal is requested (by setting the lifetime to zero as we shall see in the messages semantics section) the associated Middlebox resource should be removed.

When removing any PDNFS instances, PDI, PDO or SF, the associated message routing states should be removed as well. The decision to remove unused Messaging Associations, should be dependent on the policies discussed in Chapter 4.

5.11 Operating the PDNFS Protocol in the Path-Decoupled Mode Of Operation

As discussed in Chapter 3, when a PDSAP protocol deployments starts, application aware end-hosts might not always support the PDSAP. Alternative deployment options are possible where an application aware entity would signal the PDSAP aware network nodes on behalf of the application aware end-hosts.

Since the PDMTP and the PDNFS protocol have a clear distinction between message routing addresses and data flow addresses, the above Path-Decoupled mode of operation would be possible. Several deployment models could be envisaged:

- An application proxy would send messages to the application client address and Middleboxes on the path would intercept the message and enforce the required behavior on the data path.
- An application proxy would send messages to a specific Middlebox and in that case operate similarly to targeted signaling protocols discussed in Chapter 3. In this case the unilateral signaling mode (discussed in section 5.9.4) would always be used.
- An application proxy would send a PDTS PDSAP message to the application client address, requesting the use of the PDNFS PDSAP as discussed in section 6.1.3.

5.12 Threats Analysis

The PDNFS protocol is subject to the following threat types:

- MITM attacks categorized as follows:
  - Message type modification: the protocol operations will be completely in jeopardy
  - Parameter modification, the attack severity depends on the modified parameter:
    - a mapped address and port object could redirect data flow traffic to victim nodes or for data flow hijacking purposes
    - a message sequence change would disrupt the protocol operations as no response message could be validated
    - error conditions: minor errors could be changed to critical errors and a PDI or PDO would not retry to send its message
    - Life time duration: if set too low could induce DoS attacks on the nodes CPU due to high refreshing rates. If this parameter was set too long it could allow entities to hijack data flow sessions in case one of the application end-hosts is no longer available and the attack would have spoofed its address.
    - Nonces used for cross-referencing a PDI message with a PDO message: if this parameter is changed MAD messages would be continuously sent and PDO messages will not be forwarded beyond the Edge-NAT
    - MRSUID in MAD messages: APF messages initiated by the Edge-Middlebox will be known as initiated by the PDO instance which will induce the PDI instance to stop sending MAD message. Therefore the PDNFS SF's on all the Middleboxes will expire and the data flow packets will stop being forwarded
  - PDNFS SF ownership take over and associated impact on the data flow when correlating a MAD message with an APF message
  - Message dropping and redirection to victim nodes
  - Dropping or adding optional parameters: signaling mode in case the protocol is ran in bilateral mode, inbound packet forwarding might not work if bilateral mode was not applicable.

  - Message injection: NOTIFY and error PDRESP messages if not properly validated and if embedding critical error conditions could have dramatic disruption for the protocol

- Message floods
- Identity or role spoofing: a PDI or PDO identity or other form of credentials if spoofed could allow malicious parties to attack the network infrastructure.
- When the data source address is not known, attackers could use port scanning tools to detect a valid mapped port and existing pinhole with wild-carded data source[22] entry.

Any intermediary attacking the PDMTP discovery phase by intercepting messages and inserting itself (in the PDNFS signaling path) could be able to perform the MITM attacks that we have discussed above. Hence it is critical to be able to validate the role of an intercepting node and the validity of that role in the intercepting node's network: an entity could claim to be a Firewall and might be even be asserted to have that role but that assertion might not be valid in the network where the node is situated. In Chapter 6 we shall discuss a PKI framework to provide such an assertion based on investigations done in [19].

5.13 Authentication and Authorization Mechanisms for the PDNFS Protocol

A newly introduced protocol will only be successful if it does not induce new security attacks on the network infrastructure. The PDNFS protocol's nature requires careful attention from that perspective as the ability to allow malicious entities to use the protocol is by itself a major threat to the network infrastructure. We can split this discussion into two parts: authenticating and authorizing PDIs and PDOs and authenticating and authorizing PDNFS SFs.

[22]The data flow source address could be deduced when the remote end-host's data flow receiving socket is known, this might not always be applicable to the data flow source port although RTP and RTCP implementations are now known to use the same socket for sending and receiving

5.13.1 Authentication and Authorization of PDIs and PDOs

The PDNFS protocol leverages hop-by-hop security associations provided by the PDMTP protocol but this would not be enough to ensure a network intermediary that a PDI or PDO was successfully authenticated and authorized to request a specific NAT bind or pinhole creation. A PDI or PDO does not necessarily need to be authenticated when role-based authorization can be performed but proper assertion needs to be considered to securely bind the role with the PDI or PDO.

We have discussed several authentication and authorization frameworks in sections 3.3.5, 5.4.1 and 5.5. Leveraging the previous discussions, we could summarize the discussion as follows:

- Network deployments where PDNFS signaling messages are kept within the same network (this would be typically deployments where only unilateral signaling is always used) could use domain based authentication and authorization mechanisms. Such mechanisms include the use of the Kerberos protocol [57] with user profiles locally stored on each Middlebox or centralized in LDAP directories [80]. These deployments use hop-by-hop authentication and authorization, which implies that PDNFS SF distant by more than one PDNFS hop away can't authenticate and verify the privileges of a PDI or PDO.

- Network deployments where PDNFS signaling messages are sent between different trust domains where end-to-middle security is required without the ability to use a domain based authentication and authorization solution as is proposed in the single domain case. In these deployments Public Key Certificates [81] [55] would be required to authenticate PDIs or PDOs across trust domains, however authorization assertion (to request a pinhole or NAT bind creation) could not be necessarily provided with attributes embedded in certificates but a directory based solution for attribute verification could be used. Moreover PDNFS SFs could also rely on trusted PDNFS PDIs to approve requests sent by PDNFS PDOs hosted beyond their trust domain. These deployments rely on the use of signatures on non-mutable fields of a PDNFS message by using the Cryptographic Message Syntax (**CMS**) defined in [82], non-mutable fields are listed in section 5.14. The method described in [83] would be used to sign and protect all these immutable fields.

  The real-time impact of generating and verifying the non-mutable fields' signature needs

to be taken into account when comparing the alternative security solutions. We can estimate real-time impacts with the openssl cryptographic tool kit[23] [84] to be in the following range of

- 0.5 ms when verifying (on a PDNFS aware Middlebox) the signature on a 70% loaded 1 Ghz PowerPC processor
- 3 ms when signing the objects on a 70% loaded 1Ghz PowerPC processor

Hence the overall delay due to the object signing would be in the range of 3ms + 0.5 x number of PDNFS intercepting nodes, we could say that on the average this overall delay would be around 4.5 ms if three Middleboxes intercept the message.

5.13.2 Authentication and Authorization of PDNFS SFs

In section 6.1.4 we shall discuss the usage of a Public Key Certificate to assert the role of a PDNFS aware Middlebox in the network. Within the same trust domain we could envisage the deployment of a shared symmetric key for all Middleboxes. However this solution implies that if an attacker managed to steal the key it can impersonate a Middlebox in the network. Therefore if a symmetric key shared among all the Middleboxes were used, it could only be used for a limited amount of time before replacement. Alternatively a Kerberos [57] based solution could be used as discussed in section 5.13.1. Privilege verification could be either based on a local table (for small networks) or on a centralized database by using LDAP [80].
Message or (mutable) object integrity protection can't be easily achieved for PDNFS SFs as this would imply that the message grows continuously when SFs along the path modify existing objects (the object history would need to be kept in the message). Whereas inserted Non-Mutable objects could be signed by the PDNFS SFs. As CMS will be used, the concepts discussed in [19] could be leveraged. Messages generated by PDNFS SFs (PDRESP, APF for unilateral signaling or in response to a MAD or NOTIFY) could potentially have a signature on the non-mutable message fields (message type, sequence number, lifetime, and nonce for APF and NOTIFY messages).

[23] Tests were run with RSA signatures and with 512 bit keys

The PDNFS candidate protocols could use the semantics defined in this section, while following the traditional IETF standard 32 bit word alignment and Type Length Value field attributes for messages and objects. All non-mutable objects will be indicated with **NM** (None Mutable)and should preferably be signed when inter-trust domain signaling messages are exchanged based on the outcomes of section 5.13.1.

5.14.1 Mapped Address Determination (MAD) Message Semantics

To meet the MAD message's requirements defined in sections 5.14.1 and 5.9.4, the following information must be incorporated:

- NAT bind lifetime: set to zero to remove the NAT bind and induce removal of PDNFS SF instances created by the Edge-Middleboxes' initiated APF message. This is normally a mutable field however when the value is set to zero the object needs to be part of the signed objects as only the PDI is authorized to initiate the removal.
- Message Sequence number - NM
  - Interface address
  - Local receiving port number
- Preferred Mapped address - NM
- IP protocol type (for example: TCP, SCTP, UDP, DCCP, IPSEC) - NM
- Preferred Mapped port - NM
- Preferred Mapped address and port pair - NM
- Nonce - NM
- Signaling mode flag: Bilateral signaling or unilateral signaling, this information is optional and only used when the Bilateral signaling mode is used. NM

- Asynchronous Notification support request: this information is only required when notifications are required due to lack of application failure detection mechanisms. NM
- Port Parity preservation - NM
- Contiguous port block allocation NM

5.14.2 Allow Packet Forwarding (APF) Message Semantics

To meet the APF message's pinhole creation and maintenance requirements, the following information should be embedded in an APF message:

- Outbound data flow filter descriptor (this is sent to the PDMTP layer and sent as part of the PDMTP objects as discussed in section 4.11):
  - IP Protocol type (for example: TCP, SCTP, UDP, DCCP, IPSEC)
  - Data flow source address
  - Data flow source demultiplexer: source transport port
  - Data flow destination address
  - Data flow destination demultiplexer: destination transport port or SPI (for IPSEC)
- IPv6 flow label (if applicable): NM
- Pinhole lifetime: set to 0 to remove the PDNFS PDI and SF instances and all associated Middlebox resources. When the 0 value is used, the object should be part of the signed objects
- Message sequence number - NM
- Asynchronous Notification support: this information is only required when notifications are required due to lack of application failure detection mechanisms. NM
- NAT count object
- Contiguous port numbers: in case of data flows with contiguous port numbers, one PDNFS message could be sent requesting the required number of contiguous ports. In case

of RTP and RTCP implementations prior to [77] this value would be one (one additional flow with a consecutive port). - NM

5.14.3 PDNFS Response (PDRESP) Message Semantics

To meet the PDRESP message's requirements defined in sections 5.9.1, 5.9.4 and 5.14.2 the following data should be embedded in PDRESP messages:

- Response type: Success or Error - NM
- Message Sequence Number: NM
- Successful response objects (no object is necessary for an APF positive message response):
  - Mapped External address and port, several objects of this type could be inserted when several NATs are deployed within the same administrative domain- NM
  - Mapped Internal address and port - this pair is inserted by twice NATs - NM
  - MRSUID: used only during unilateral signaling operation to inform the PDI of the MRSUID that would be used to route APF messages sent by the Edge-Middlebox - NM
  - Last PDNFS aware hop: this flag is set when a PDNFS aware node detects that it is the last downstream node on the path - NM
- Accepted lifetime
- Error object types:
  - Authorization error values (all NM) - persistent error condition:
    - PDI unauthorized to request for a NAT bind
    - PDO unauthorized to request a pinhole for a specific component of the data flow descriptor
  - Resource error values (all NM) - ephemeral error condition:
    - Mapped address pool is exhausted

    - Mapped port pool is exhausted
    - Requested mapped address and port pair is already allocated
    - Unavailable resources: memory or processing power
  - System issues (NM)- persistent or ephemeral errors: host under maintenance, kernel panics etc ...
  - Requested pinhole lifetime is unacceptable (NM)

5.14.4 NOTIFY Message Semantics

When requested by a PDI or PDO during the instantiation of a PDNFS SF or after cross-referencing an APF to an existing PDNFS SF, the PDNFS SF in question would transmit a NOTIFY message which would include the following information:

- Error object types - all NM:
  - Unavailable resources -ephemeral error: memory or processing power
  - System issues - persistent or ephemeral errors: host under maintenance, kernel panics etc ...
  - End-to-end path-discovery is required - persistent error

5.15 Summary

In this chapter we have described the various options for path-directed NAT and Firewall signaling protocols while demonstrating that a single combined protocol should be used for both NAT and Firewall signaling. Furthermore the mode of operations and semantics of the proposed protocol framework allows the specification of a protocol that would be used for both path-coupled and path-decoupled signaling. The proposed framework allows a smooth transition towards an Internet where Path-directed NAT and Firewall signaling would be used without imposing constraints on the various applications to support capability exchanges. A unilateral scheme is used where the end-hosts supporting the proposed framework acts as if the remote end-host is not PDNFS aware. Then if the remote end-host turns out to be PDNFS aware, the

protocol framework provides for fall-back to a bilateral signaling scheme. The proposed unilateral scheme is also used when the protocol framework is deployed in path-decoupled scenarios. If not deployed with the appropriate security measures discussed in sections 5.4.1, 5.5, 5.13 the protocol framework could be used by malicious parties to attack the network infrastructure. Accordingly, the proposed security mechanisms are mandatory when deploying this technology in secure network environments.

Chapter 6

Engineering considerations for PDNFS deployments in the Internet

We have discussed in Chapters 3, 4 and 5 the framework for a Path-Directed NAT and Firewall signaling technology with in depth coverage of the required mode of operations and protocol semantics.

In this Chapter we shall consider how to apply that technology in live network deployments where users are running their day to day activity and assess the required engineering considerations.

In section 6.1, we analyze deployment planning and infrastructure impacts when deploying the PDNFS protocol.

In section 6.2, we discuss the use of the PDNFS protocol in IPv6 networks where IPv4 NAT deployments are no longer applicable and validate the protocol in NAT free environments.

6.1 Deployment Planning and Infrastructure Impact Assessment

The decision to deploy a new technology in a live network with potential loss of revenue streams is always a challenge for Chief Information Officers.

For the PDNFS protocol, deployment planning is a critical phase. It is essential that the network security infrastructure gets upgraded first (as we shall see in section 6.1.4), followed by Middleboxes, then by application proxies and finally ending with end-hosts upgrades. If this upgrade order is not followed users and applications may not take advantage of the new protocol, and the application aware entities will continue to use the existing NAT and Firewall

traversal mechanisms with their known applicability issues.

There are several steps required to plan the deployment of the PDNFS technology:

- Create an inventory of all the applications used which are impacted by NATs and Firewalls based on the analysis carried out in sections 2.4.2 and 2.4.1.
- Create an inventory of all the end-hosts that use the applications listed in the inventory in item 1 and determine if they can be upgraded with new software providing them with PDNFS protocol support. The inventory should consider the associated timeframes in which the various equipment vendors are expecting to support the PDNFS protocol.
- Create an inventory[1] of Middleboxes deployed in the network as well as the application proxy nodes associated with the affected applications.
- Plan the roll-out of the extended public key certificate enrollment components discussed in section 6.1.4 as well as other network security key constituents required for using PDNFS (discussed in section 6.1.4)

In the event that application end-hosts' PDNFS support will be available too late or it proves to be impossible to add PDNFS support to some end-hosts, the application proxies could be upgraded to support the PDNFS protocol and operate using the path-decoupled operation as discussed in section 5.11 with or without the use of the PDTS PDSAP discussed in section 6.1.3. However as we shall see in section 6.1.3, the PDTS PDSAP requires certain network upgrades on the stub networks routers; these upgrades need to be considered as part of the above deployment project planning.

6.1.1 Performance Potential Issues and Proxying

In Chapter 4 we have discussed how PDSAP messages are routed towards their destination through the creation of a sequence of Messaging Associations between pairs of discovered PDSAP aware neighbors which are one PDMTP hop apart.

The Messaging Association establishment has non-negligible real-time impacts on applications.

[1]The inventory will indicate the timeframes in which both the Middleboxes and application proxies vendors are expected to support the PDNFS technology

We summarize these impacts in Table 6.1; however the real-time delays vary depending on the Round-Trip Time (**RTT**).

In section 4.5 we discussed proposals to avoid the MA establishment delays by maintaining idle MAs for a certain period after they were no longer in active use. These MA caching proposals would only be applicable if the PDMTP nodes had already established MAs due to prior message exchanges. Since the purpose of the PDNFS protocol is to allow application

Table 6.1: Messaging Association types' delay assessment

Nb Messages	**IPSEC/TCP**	**TCP/TLS**
PDMTP no MA	2 (1 RTT)	2 (1 RTT)
IKE exchanges	6 (2 RTT)	NA[2]
TCP connection	3 (1.5 RTT)	3 (1.5 RTT)
TLS exchanges	NA	6 (3 RTT)
Total MA msgs	9 (3.5 RTT)	9 (4.5 RTT)
Total PDMTP delays	11 (4.5 RTT)	11 (5.5 RTT)

packet traversal from a data sender to a data receiver, the signaling messages are potentially sent across different networks. As a result the MA caching methods discussed above would not be applicable as it would not be possible to have pre-established MAs with all the networks connected to the Internet.

We can estimate the MA establishment time based on using average RTT times and RTT limits based on the application type.

If we consider the 150 ms one way delay limit for VoIP calls[3], the maximum acceptable RTT would be in this case 300 ms. Assuming two to four Middleboxes are deployed between both users (with an even 50 ms or 30 ms[4] for each hop), the RTT used for calculating overall MA establishment duration would be 100 ms (2 Middleboxes) or 60 ms (4 Middleboxes) this would bound the MA establishment delay to 700 ms (2 Middleboxes) or 480 ms (4 Middleboxes)[5].

This is definitely problematic for VoIP users as, based on Table 6.1, the overall PDMTP

[2]NA: Not Applicable

delays could be up to 2.1 to 2.4 seconds (in the worst case) in total. After adding the PDNFS protocol round-trips and processing time, the overall solution delay could easily go beyond the 3 seconds. This would definitely be a major issue for PDNFS deployments in the Internet.

Due to the impacts of the calculated delay times on real-time applications, we propose to use the PDNFS protocol primarily within an administrative network where in the worst case two Middleboxes would be deployed if we consider deployments similar to the one described in Figure 2.5.

The Edge-Middlebox will always respond to PDNFS messages (not only to the MAD messages as done in the unilateral signaling mode) as if there was no PDNFS downstream node. This would bound the PDNFS and PDMTP delays to the PDMTP delays and 2 RTTs (for MAD and APF messages) added to the PDNFS message processing for the MA and APF messages. These delays only occur for the establishment of the MAs as the lifetime of the MA will be long enough to limit the number of MA (based on our discussion in section 4.5) establishment. We show this mode of operation in Figures 6.1 and 6.2, where not all the PDMTP and PDNFS messages are shown (MRS-EST, MA establishment, PDNFS MAD refreshes) for readability purposes.

If we apply the MA establishment delay calculation method (Table 6.1) with an RTT of 15 ms between all the interacting nodes (hence a total of 30 ms RTT between PDI/PDO and NATFW2) the overall delay to get a bi-directional data stream could be calculated as follows:

- Message routing state creation for MRSUID1 with MA1 (TLS) establishment duration: $5.5 * 30ms = 165ms$
- 1 PDNFS RTT (between PDI and NATFW2) = 30 ms[6]

[3]Real-time applications such as VoIP require a maximum one way delay to ensure full duplex calls. If the one way delay exceeds 150 ms users could talk at the wrong time as they wouldn't have heard their correspondent

[4]3 MAs establishments are needed when 2 Middleboxes are on the path, 5 MAs establishment are needed when 4 Middleboxes are on the path

[5]Since cryptographic operations are within the 10 ms processing time range, 150 ms for these operations is a realistic value

[6]The SIP invite message could be sent after that time (around 200 ms with the cryptographic calculation), simple[7] ring-back tone could be heard by the calling party as soon as the SIP RINGING message is received but that delay is not attributable to the PDNFS protocol

[7]In some cases users could get a specific announcement that wouldn't be generated locally on the end-host,

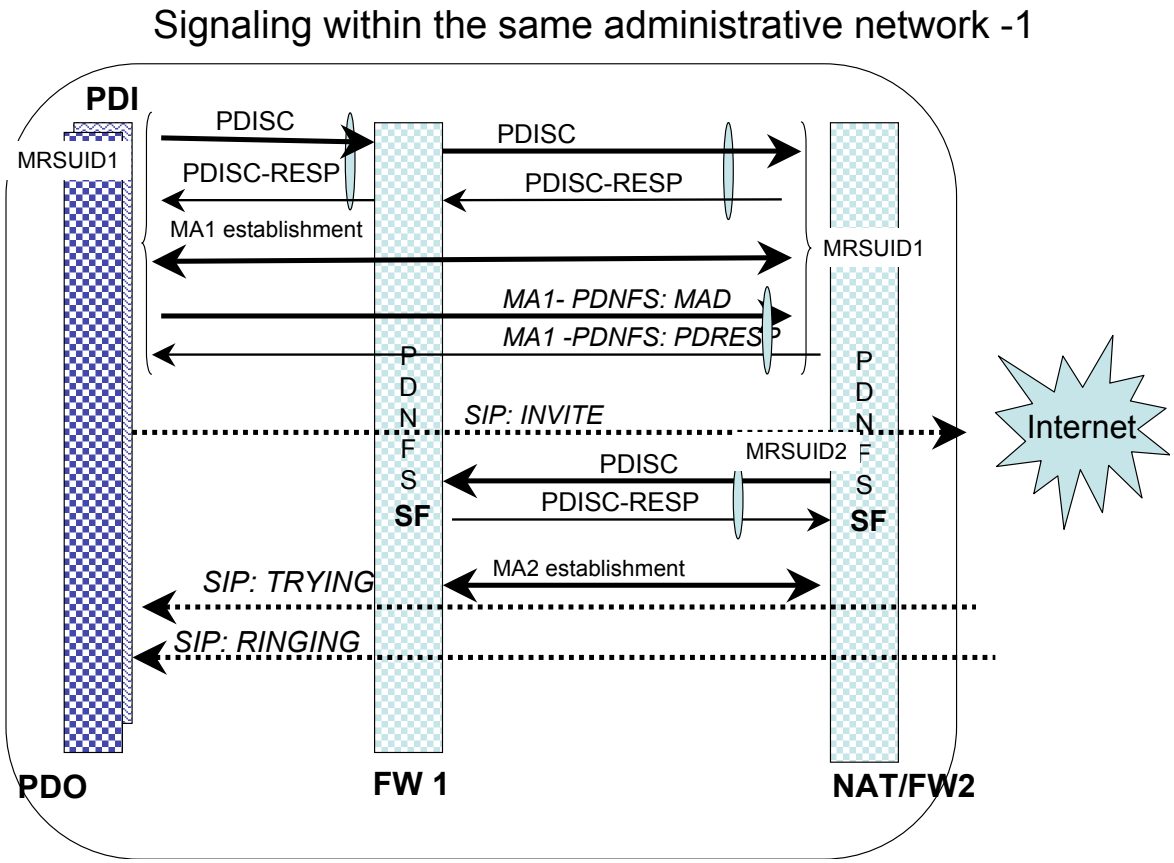


Figure 6.1: Minimizing the PDNFS protocol exchange delays with single administrative domain signaling

- Message routing state creation for MRSUID2 with MA2 (TLS) establishment duration: $5.5 * 15ms = 82.5ms$
- 0.5 PDNFS RTT (between NATFW2 and FW1) = 7.5 ms
- Message routing state creation for MRSUID2 with MA3 (TLS) establishment duration: $5.5 * 15ms = 82.5ms$
- 1 PDNFS RTT (between PDI and FW1) = 15 ms
- 0.5 PDNFS RTT (between FW1 and NATFW2) = 7.5 ms[8]
- Message routing state creation for MRSUID3 (MA3 is reused): 1 RTT = 15ms

and that announcement can only be heard once inbound packets can be sent to the end-host

[8]The user could start listening to a specific announcement starting that moment

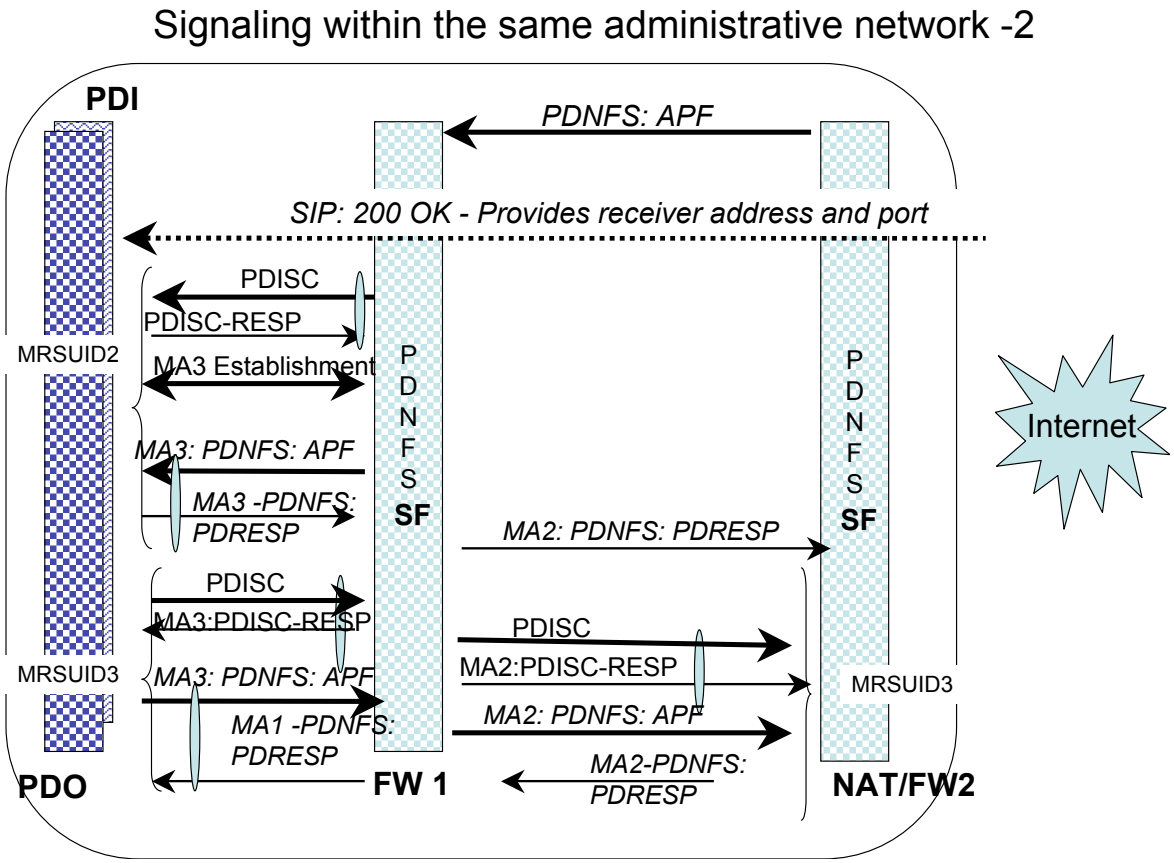


Figure 6.2: Minimizing the PDNFS protocol exchange delays with single administrative domain signaling: part2

- 0.5 PDNFS RTT (between PDO and FW1) = 7.5 ms
- Message routing state creation for MRSUID3 (MA2 is reused): 1 RTT = 15 ms
- 0.5 and 0.5 PDNFS RTT (between FW1 and NATFW2) = 15 ms
- 0.5 PDNFS RTT (between FW1 and PDO) = 7.5 ms

The total delay without taking into account cryptographic calculations is about 450 ms, with the cryptographic calculation a realistic overall duration would be about 550 ms. This limit is still acceptable for real-time applications such as VoIP where ring back tone is expected to be heard by the calling party after at most two seconds. In the above calculation simple ring-back tone could be heard at about 400 or 500 ms, the PDNFS protocol contributes about 200 ms to that delay; whereas the remaining delay is due to the VoIP protocol itself, the 200 or 300 ms

could be realistic values for the SIP protocol considered in the example studied.
The bi-directional packet flow traversal would be enabled after 550 ms which is an acceptable delay, and pinhole tuning messages would be used once the correspondent end-host responds with the 200 OK message (the tuning would be done with a MAD message sent by the PDI existing instance) but that message processing does not affect the user as the data packets are already flowing.

Average sized and big networks are multi-homed to ensure a highly available Internet connectivity, these networks could suffer from routing asymmetry problems where outbound packets could be sent through one Firewall and inbound packets by another Firewall. These routing asymmetry problems affect our PDNFS deployments when NATs are not deployed. This would be an issue to solve particularly for IPv6 networks (discussed in section 6.2) which are not expected to deploy NATs but will be expected to make extensive use of signaled Firewalls once end-to-end IPsec becomes widespread.
In Figure 6.3, we describe the routing asymmetry problem as it impacts the PDNFS proxying solution and show that it has acceptable MA establishment times. In Figure 6.3, the PDI has triggered one of the Edge-Middleboxes (EM-FW1) to send an APF message to allow packet traversal for the data flow. However data flow packets are reaching a different Edge-Middlebox (EM-FW2), hence the inbound data flow packets will be dropped by EM-FW2. We discuss our proposal to solve that problem in section 6.1.2.

6.1.2 Solving Asymmetric Routing Issues

Asymmetric routing has always been known to create problems to network administrators and is not specific to the PDNFS protocol.
Existing networks deploying several stateful Firewalls at the edge of their networks, have already been impacted by such problems:

- Inbound UDP datagrams might not traverse the same Firewall as outbound UDP datagrams which have created a pinhole on the stateful Firewall.
- Inbound TCP SYN ACK might not traverse the same Firewall as outbound TCP SYN of the same TCP connection which created a pinhole on the stateful Firewall.

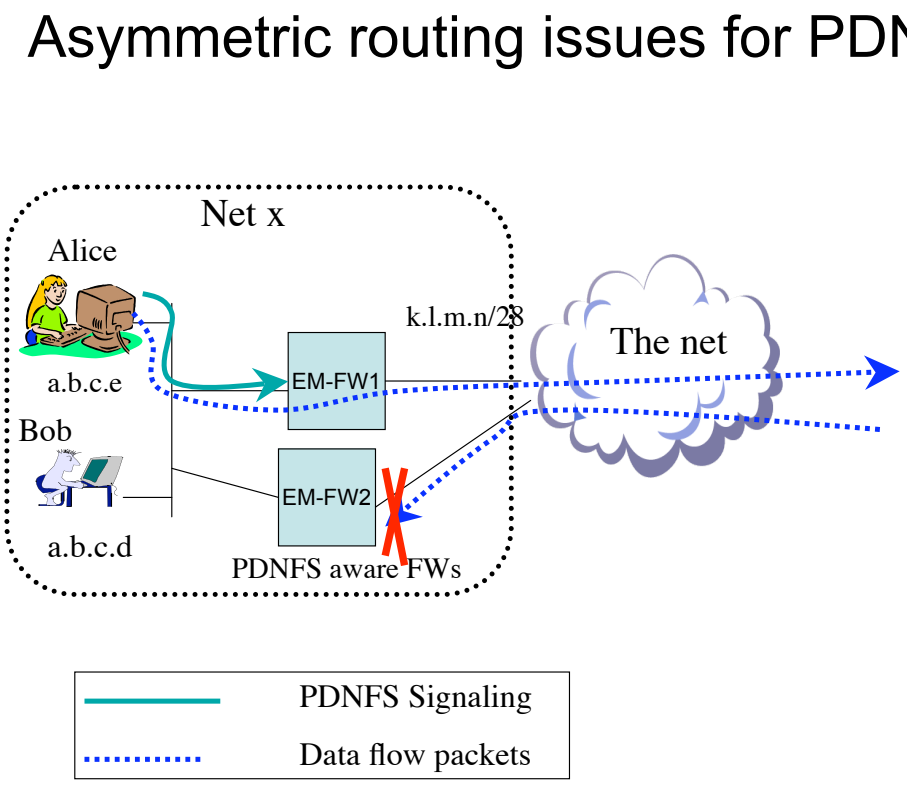


Figure 6.3: Asymmetric Routing issues

In [18], we have analyzed in detail the origins of asymmetric routing and its impact in multi-homed infrastructures using network perimeter Firewalls and have compared the possible solutions to the problem. The outcomes of the analysis identifies the Firewall state synchronizing method as the best solution.

Firewall state synchronizing consists of replicating the various Firewall states related to individual flows between the Firewalls protecting a domain. Existing Firewall state synching implementations[8] suffer from a high overhead traffic (estimated to be up to 50% depending on the data traffic type).

We have proposed in [18] a method which has specific state synchronizing mechanisms based on the traffic type (ICMP, UDP, TCP and IPSEC). Our simulations indicated that our proposed

[8]Currently there is no standard for Firewall state synching, all implementations use proprietary systems, only CARP [85] and PFSYNC [21] based solutions have publicly available specifications.

improvements made to PFSYNC [21] (used implementation by the OpenBSD PF Firewall) can minimize the state synching traffic and limit it to 5% of the overall data flow packets.

Our approach could be used in other Firewall state synchronizing implementations, further investigation is required to analyze the integration of our work in the IETF Context Transfer Efforts (although that work is currently mainly focused on seamless mobility).

6.1.3 A Proposal for a Path-Directed Trigger Signaling Protocol

Throughout this dissertation we have discussed the Path-Directed Signaling protocol suite concepts and its applicability in the Internet. We have discussed the use of the protocol in both path-coupled and path-decoupled modes.

In sections 4.9.2 and 5.11, we have identified the usefulness of a framework in which an application aware entity would trigger network intermediaries on the same LAN segment as the (PDSP suite unaware) application end-hosts to initiate specific PDSAP signaling exchanges to enforce behaviors on the data flow packets.

When particular application end-hosts cannot be upgraded on time or their upgrade will never be possible, the use of Path-Directed signaling could still be possible while signaling the appropriate network intermediaries.

In [20], we have provided a framework where a Path-Directed Trigger Signaling (**PDTS**) protocol could be used where the application end-hosts' on-link routers would be upgraded to support the PDMTP layer, the PDTS PDSAP as well as the other required PDSAPs (including our PDNFS protocol).

Our PDTS PDSAP proposal's impacts on the deployment of the PDNFS and PDMTP can be summarized as:

- On-link routers will need to support the PDNFS protocol and provide appropriate credentials (the PKC extensions discussed in section 6.1.4 could be used for that purpose) asserting their role in the network infrastructure

- The application proxies acting as the signaling initiators for PDTS will need to provide the appropriate credentials to the on-link routers

- The application proxies will have to be in a different subnet than the application end-hosts (as the PDTS signaling messages need to be intercepted by the on-link routers)

- The application proxies need to be aware of which end-hosts need their support in signaling the network infrastructure for NAT and Firewall Traversal. In [20], we discussed the usage of a specific network signaling parameter in the application protocol to indicate the support of the PDNFS protocol.

  Alternative solutions could be used for that purpose where the application proxy would query a database based on the user profile, however these solutions are not always applicable when users login from different end-hosts which might have completely different capabilities (some of them could support PDNFS, others might not).

6.1.4 Network Security Infrastructure Impacts

The deployments of PDNFS adds new requirements for the network security infrastructure, we shall classify these requirements as follows:

- In simple network deployments a single public key/private key pair or a symmetric key could be used by all Middleboxes (this was considered as the simplistic solution for PDNFS SF authentication). In complex networks this simplistic view might not be applicable and Middleboxes would require credential enrollment as well as credential validation. We discuss these requirements in section 6.1.4.1

- Authentication and authorization mechanisms for end-hosts using the PDNFS protocol: we discuss these requirements in section 6.1.4.2

6.1.4.1 Usage of Specific PKC Extensions to Secure the Deployments of the PDNFS Protocol

Due to its topology unaware nature, the Path-Directed signaling protocol suite could be subject to attacks during the neighbor discovery phase where a network intermediary could claim to be a valid intermediary for enforcing the requested behavior on the data flow packets.

In [19], we proposed to use specific Public Key Certificate (**PKC**) extensions asserting the capabilities of a network intermediary and identifying the networks in which that intermediary is authorized to enforce its supported capabilities. The proposed PKC content is shown in Figure 6.4 and leverages existing work to assert network prefixes to routers [86].

Our solution described in [19], consists of using the PKC shown in Figure 6.4 and the appropriate solutions needed to secure:

- Middlebox certificate enrollment
- Delivery of the Certificate Authority's root certificate [55] to all the relying parties [55], this would include end-hosts instantiating PDIs and PDOs.
- Certificate validation

With our work in [19], Middleboxes and on-link routers (used in our PDTS proposal [20] have the appropriate credentials asserting their role in the network infrastructure where they reside. These certificates will be used during the establishment of IPsec- or TLS-based Messaging Associations. Edge-Middleboxes will use these certificates to sign the PDNFS's APF message objects when using CMS [82] as described in section 5.13. When the PDNFS NOTIFY messages are used, the Middlebox notifying the PDI or PDO MUST use its PKC to sign the NOTIFY message fields as described in section 5.13.

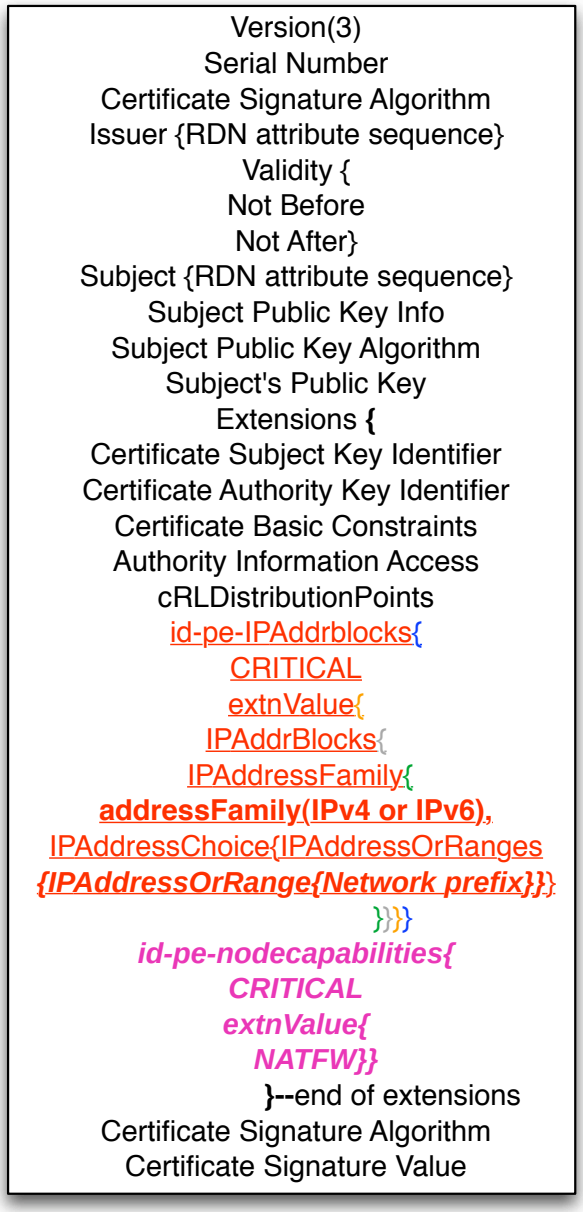


Figure 6.4: Proposed PKC for Middlebox capabilities assertion

6.1.4.2 Authentication and Authorization Infrastructure Requirements for PDNFS Aware End-Hosts

In section 5.13, we discussed two approaches for authenticating and authorizing PDNFS aware end-hosts. One approach was applicable to a single authentication realm (which would be applicable to the proxied mode of operation discussed in section 6.1.1), the other method was applicable across several authentication realms.

The first method using Kerberos' [57] trusted third party authentication would require the following:

- A Key Distribution Center (**KDC**) (integrating the Authentication Server - **AS** - and the Ticket Granting Server - **TGS** - as specified in [57])
- Mechanisms for an end-host to discover its realm's KDC, these mechanisms could include usage of appropriate DHCP options, static configuration (which wouldn't be user friendly) or usage of DNS SRV records [87]. In addition the end-host has to learn its principal name (DHCP options could be again used).
- A symmetric key (the master key) stored on the KDC for the PDNFS user (assigned with a specific Kerberos principal name within the Kerberos realm assigned to the administrative network)
- The end-host requests from the KDC a Ticket Granting Ticket (**TGT**) and then the end-host uses the TGT to request a session ticket to get authenticated by the network's Middleboxes prior to any Middlebox interaction (to avoid real-time impacts). This implies that Middleboxes deployed in that network either use the same public key/private key pair[9] or the same symmetric key. When the public key/private key pair is used the PKINIT [88] extensions to Kerberos must be applied.
- User privileges are stored either on all the Middleboxes or in a centralized repository such as an LDAP [80] directory

[9]In case the PKC discussed in section 6.1.4.1 is used by Middleboxes, all the PKCs allocated to every Middlebox in the same administrative network would have the same public key. In case one Middlebox was subverted it could impersonate other Middleboxes

The second method requires the use of a PKC for each end-hosts, although PKI have been deployed in the Internet for more than 15 years (mainly due to Internet commerce) it has still not made its way into corporate networks' desktops. One of the issues with PKI deployment for desktops and laptops is the necessity of PKI maintenance especially for technology challenged persons.

If a certificate's private key is stolen, the attacker could impersonate the certificate's user. Certain users might not be aware of all the dormant processes run by all their shareware and freeware applications which could retrieve their private keys. Even if the user was made aware of his problem, not all users will contact their IT department to revoke the certificate and request a new one. Because of all these issues (which are not technology issues but rather human behavior factors), PKI has not yet generally made its way to the corporate desktop.

We have investigated the use of PKI for authenticating PDIs and PDOs instantiated on desktops and we could summarize the infrastructure requirements as follows:

- A certification authority, with all the associated tools to create certificates for every end-host as well as the required processes to manage the Certificate Revocation List [81] [55]
- End-host privileges can't be stored in their associated PKC as this would limit the PKC's lifetime and would increase the operational effort of IT departments. Hence we propose to use a directory (with LDAP [80]) based solution where every end-host is indexed based on the end-hosts subject name enclosed in its certificate.

6.2 Applicability of PDNFS to IPv6 Networks

The quest for a next generation Internet Protocol to handle a bigger addressing space than IPv4 started in the early 1990s. IPv6 has almost reached a mature stage and is certainly no longer just a toy used in research labs. IPv6 deployments have started in Japan and Korea, and several other countries. IPv6 is on almost all the desktop and laptop operating systems now in common use (Microsoft's Windows XP®, Linux 2.6 Kernel, MAC OS X® and all the BSD variants).

Many Chief Information Officers (CIOs) believe that even with a large addressing space

in their hands, an address translation system would still be required to protect their network infrastructure because marketing messages have incorrectly equated the use of NATs with increased network security. Other CIOs still think that IPv4 networks could communicate with IPv6 networks but using the **NAT-PT** [89], but this basically disregards the recommendations of [40].

Furthermore we have documented in [5] a detailed analysis of NAT-PT issues and reasons to deprecate its use and limit it to very limited deployments where application proxies, IPv6/IPv4 (or IPv4/IPv6) tunneling or dual stack nodes would not be suitable .

The authors of [90] clearly explain the mechanisms available in IPv6 networks to protect the network infrastructure. The above brief discussion is to highlight that IPv6 network deployments are no longer a "potential" next step for the Internet. Instead these deployments are real in some small parts of the Internet and they are growing.

Our work is fully compatible with IPv6:

- The PDMTP protocol could be used over IPv4 or IPv6
- Data flow descriptors could include IPv4 or IPv6 addresses as well as IPv6 specific fields such as the flow label
- The data flow descriptors could also include an IPsec SPI and IPv6 local and remote addresses of the SA end-points. As most IPv6 deployments seem to encourage the use of end-to-end IPsec SAs (versus the use of IPsec tunnels to IPSEC gateways), Firewalls would not be able to filter based on transport port numbers. Hence the PDNFS protocol allows to signal the Firewall to allow packets with specific IPsec SAs identified each by the triplet constituted by the SA end-points addresses and the receiving side SPI.

When the PDNFS protocol is used in IPv6 networks interconnected via IPv6/IPv4 tunnels, these tunnels operate as serial links and would not require any specific behavior from the protocol.

6.3 Summary

In this chapter we have analyzed the performance of the PDMTP and PDNFS and its impact on real-time applications. The result of this analysis indicated that round-trip times may have a severe impact on the protocol's performance; this implied that PDNFS signaling messages have always to be proxied at the edge of the network (unless the overall delay between the end-hosts is sufficiently small).

The required proxying creates new issues due to routing asymmetry (when applicable), for which we have proposed to use a novel Firewall state synchronizing mechanism which reduces the state synchronizing load to no more than 5% of the overall transported data traffic.

Our analysis of the security infrastructure's requirements resulted in two main solutions one using a public key certificate embedding capability assertion information for Middleboxes and another simpler method for authenticating Middleboxes by using symmetric keys (their authorization is implicit when they have the symmetric key). The end-host authentication and authorization could be achieved by either using PKCs and a directory infrastructure to handle the end-host privileges or symmetric keys with the directory infrastructure for privilege management.

Since a lot of corporate networks, where Microsoft Windows is the predominant desktop operating system, have been investing in Active Directory[10] [91] we envisage that the symmetric key approach assisted by LDAP directories would be the one to be used initially.
We have also discussed the support of IPv6 in PDMTP and PDNFS. Although IPv6 deployments are still scarce, we believe that these deployments will start soon at a large scale. The features of PDNFS will be useful in these deployments particularly the pinhole opening for data flows identified by their flow label or their IPsec SPI and the associated IPsec SA's end-hosts addresses.

[10]Microsoft's user authentication and authorisation system based on Kerberos and LDAP

Chapter 7

PDNFS Implementation Examples

In this chapter we shall define software architecture models to integrate the PDMTP and PDNFS protocols on UNIX[1] based Internet hosts, which might be either a Middlebox or an application aware end-host.

Before proceeding with our implementation examples we shall have a brief discussion on the UNIX systems properties that will be used in this chapter: Access Control.

7.1 Overview of UNIX Access Control Mechanisms

Due to the potentially dramatic impact of malicious use of the PDNFS protocol, access to the PDNFS layer on a system needs to be restricted to specific applications. UNIX systems are well known for their access control mechanisms (defined in [92]):

- A user account has access to its own files and those belonging to user groups of which the user is a member.
- A super user account ("root") has access to all files and directories.
- The access rights for every file or directory are encoded on 9 bits (rwx rwx rwx), where r is for read, w for write and x for executable. The split between the three groups of rwx is as follows:
  - The first "owner" rwx triplet (from left to right) determines accessibility for the file

[1]We have selected to analyze implementations on UNIX based operating systems due to their publicly available specifications and robustness

owner (we will ignore the differences between directories and plain files for simplicity, distinction between a file and a directory is made by an extra bit)

* For r=1, w=1 and x=1 (111), the file owner can read, write and execute the file
* For r=1, w=1, the file owner can read and write the file (the file is not executable)

– The second "group" rwx triplet (from left to right) determines accessibility for the file owner's group members. A drop box directory would typically allow members to write only, hence the triplet would have the following 011 as its value[2]

– The last "world" rwx triplet determines accessibility for any user not part of the owner's group. A drop box directory for any user would typically have the triplet set to 011

In addition the UNIX system allows delegation of authority where a specific type of users ("sudoers") could act for a specific purpose as the root (or any other permitted) account user[3].

The programs executing in a UNIX system, each operate in a process context which owns resources such as system memory and communication streams attached to devices and files in the system. Processes are owned by a user and the ownership of the process controls what resources the process can access. Processes other than those owned by the super user are strictly limited as to what resources they can access in line with the permissions associated with these resources. The relevance to our system integration work is obvious:

- The PDNFS and PDMTP processes should be owned by the root account with limited access by users: only authorized users (or only the root account could install applications that could interact with PDNFS and PDMTP)

- Applications can not send requests to the PDNFS or PDMTP unless they are being run by an authorized user account. However it is not acceptable that day to day applications run only in the root account as this would have security impacts on the end-host and the network. Hence the ideal proposal would be to have a system file, owned by the root account, in which all the authorized applications are listed. The installation of these

[2]Access to a directory implies that the directory is "searchable" by the user account based on the x bit setting

[3]The user account should be listed in the sudoer file owned by the root account

applications could only be done by the root account user (or a sudoer), hence the name and file path of the application's executable file would be included in that authorized applications list. These applications could then run in any user account without impacting the end-host's system security[4].

7.2 Integration of PDNFS with OpenBSD's PF

We have selected the OpenBSD PF Firewall as an example to show how the PDMTP and PDNFS protocols could be integrated into an operating system and a firewall sub-system which it uses; we believe the same software architecture concepts are applicable to other Firewalls such as IPFilter (NetBSD), Netfilter (Linux) or other proprietary Firewalls. We have limited our analysis to a high level integration model. Further refinements will be required to this work for a wide deployment of PDNFS with PF.

[93] provides a good overview of the PF Firewall which also provides an integrated NAT daemon. Due to the tight coupling of the Firewall (in our own terminology where a Firewall is a stateful or stateless packet filter) with the NAT daemon we shall consider that all PDNFS messages are intercepted by the system hosting PF.
For further information on PF and its basic commands, the reader is invited to refer to [94].

7.2.1 Software Architecture Proposal

In Figure 7.1, we show the various interacting functional blocks. We shall only discuss new functional blocks as the PDMTP and PDNFS functions have been described earlier in Chapters 4 and 5:

- Authentication functional block
- Authorization functional block

Although not shown in Figure 7.1, the authentication functional block plays an important block in the figure but cannot be represented as a single functional component as it could be spread on a UNIX system.

[4]Assuming that the application has no software flaws

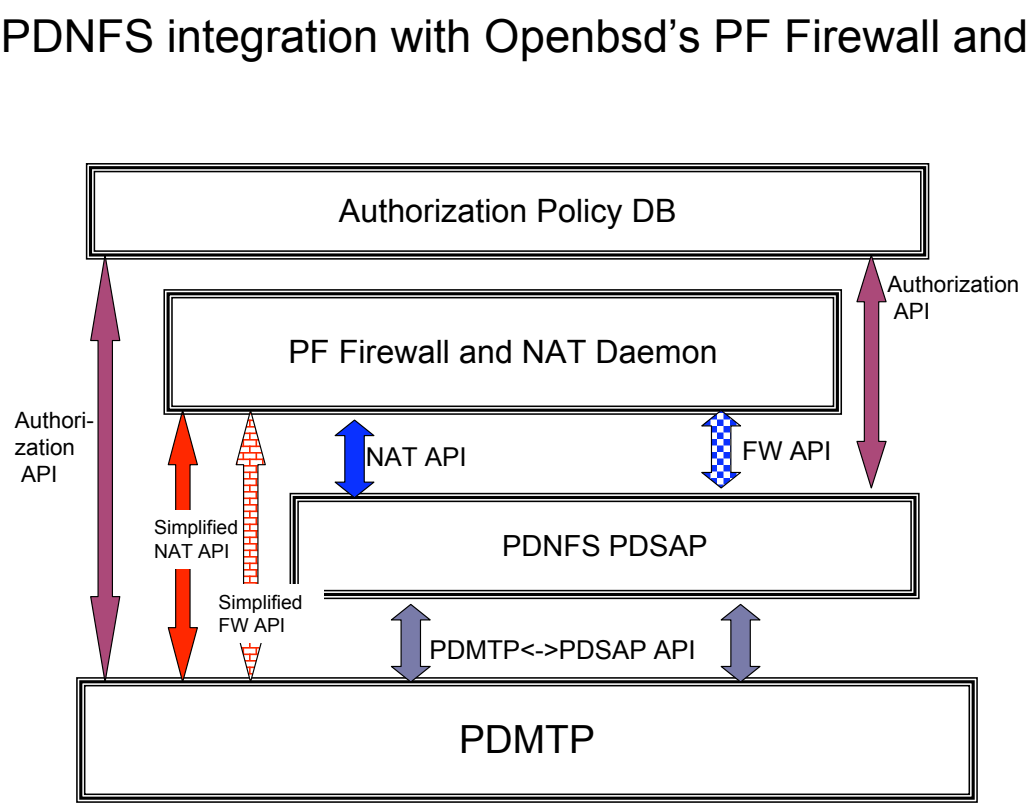


Figure 7.1: PDNFS Integration with the PF Packet Filter

Depending on the authentication approach selected, different authentication frameworks could be used:

- The **GSS-API** (Generic Security Service API) [95] with [96] when Kerberos is used to authenticate an end-host identified by its Kerberos Principal name
- The SSL API when TLS is used (including when Kerberos is used [97]or pre-shared keys with [98])
- PFKEY socket API for IPSEC/IKE Security Associations (briefly discussed in Chapter 21 of [99])

As of today the PF packet filter does not include any authorization database[5]. The same caveat is applicable to the Linux Netfilter Firewall.

We propose to create an authorization table on every Firewall using PDNFS. The authorization

table will be queried against the PDNFS user's identity (identity mapping will be required between the Kerberos' principal name or a PKC's subject with the identity for which privileges are being queried).

We envisage three types of authorization database:

- A local database manually configured (and maintained) by the Firewall's administrator.
- A local database configured by adhoc downloads based on received PDNFS requests. This type of database requires the use of an authorization server from which the Firewall will download the credential and associated privileges to be validated. Most corporate infrastructures use RADIUS, DIAMETER and COPS, hence these protocols could be used in the near future for such information downloads. A key requirement for such protocols is the ability to keep track of the downloaded policies and to synchronize[6] the database entries with the authorization server.
- A local database caching entries based on previous queries to a directory server configured with privileges against a user entry (we have discussed in section 5.13.1, the use of LDAPv3 [80] for such purposes).

7.2.1.1 PDMTP/PDNFS API's Primitives Definition

Based on our analysis in Chapters 4 and 5 of the PDMTP and PDNFS interactions, we believe that four primitives are sufficient for the PDMTP$<->$PDNFS API:

- sendmsg(RAO, PDSAP ID[7], MRSUID[8], MTR, IP version, Message Target address, Message Source address, Message IPv6 flow label, Message DSCP value, Data flow IP version, Data source address, Data sender address, data source demultiplexer[8], data destination demultiplexer[9], nonce, rediscover)[10]. This primitive is sent by the PDNFS (or other PDSAP) layer to the PDMTP layer. All the sendmsg primitive's parameters were discussed in great detail in Chapters 4 and 5. The rediscover flag is to indicate that peer

[5]We have not considered the Authpf [94] related table as suitable for that purpose as it was too specific to Authpf

[6]Although RADIUS does not support the synchronization requirement, we foresee its use in corporate networks due to its existing presence for remote authentication on network infrastructure products.

discovery is required even if an MRS entry existed for the specific MRI. The reader is invited to refer to those chapters for the use of these parameters.

- receivemsg(RAO, MRSUID, MTR, IP version, Target address, Source address, Message IPv6 flow label, Message DSCP value, Data flow IP version, Data source address, Data sender address, data source demultiplexer[9], data destination demultiplexer[9],). This primitive is sent by the PDMTP layer to the PDNFS (or other PDSAP layer). All the receivemsg primitive's parameters were discussed in great detail in Chapters 4 and 5. The reader is invited to refer to those chapters for the use of these parameters. The use of the RAO parameter is not yet defined, its presence is intended to provide a degree of future proofing in case we find a use case for it.

- clearmrs(MRSUID), this primitive removes any PDMTP instances created by the PDNFS layer for the MRSUID (PSI or PSR instances)

- error(MRSUID, errortype), we currently envisage the following error types and values:

  - No error found - provides feedback on a successful message transmission (equivalent to UNIX error code 0)

  - Message could not be delivered:

    * No response

    * ICMP error: host unreachable

    * ICMP error: port unreachable (end-host does not support the protocol)

    * ICMP error: administratively prohibited (a Firewall dropped the message)

[7]The PDSAP Identifier is required to use the API with any other PDSAP than PDNFS

[8]We believe that it is best to create the MRSUID at the PDSAP layer so as to use the MRSUID as a handle to remove PSI and PSR related to that MRSUID. Since MRSUIDs are expected to be created at the PDSAP layer, the MRSUID need be selected randomly from a large naming space. ([17] has used a randomly selected 128 bit field to ensure its uniqueness across a potentially large set of sessions. We believe that a 1 in 2^{128} collision probability as an acceptable risk.

[9]The demultiplexer used at the destination would either be in the form of a transport port or any other form such as an SPI that uniquely identifies the data flow.

[10]We have used for the MRI: the signaling message source and target address as well as the IPv6 Flow label -when applicable - and the DSCP value, other fields could be added for the MRI

        * Wrong nonce
    - MA establishment failure:
        * Failed to have common MA protocol stack
        * Authentication failure:
            · Could not authenticate discovered neighbor
            · Discovered neighbor was not able to authenticate the local host
            · Discovered neighbor does not have the credential to operate as a Middlebox (this is in reference to section 6.1.4.1)
    - Upstream neighbor has changed
    - Downstream neighbor has changed
    - Existing MA end-point neighbor is no longer reachable

As noted above, the MTR is provided to the PDMTP layer without any specific information about the protocol stack to be used, this is done intentionally as discussed in Chapter 4 to allow the use of new transport protocols in the Internet.
Since the use of Kerberos and PKC based authentication is simpler with TLS[11] we have preferred to prioritize its usage over IPsec and IKE.
The PDMTP layer would have an MTR table configured with the following MTR entries and the associated protocol stack:

- Reliable and ordered delivery, congestion control: TCP[12]
- Reliable and ordered delivery, congestion control and message integrity and confidentiality protection: TCP/TLS

7.2.1.2 PDNFS/NAT Daemon API's Primitives Definition

Prior to using primitives with the NAT daemon (in our example the NAT daemon is merged with the Firewall daemon, two APIs are used for generic purposes), the PDNFS should have

[11]The use of the TLS API is simpler than the use of PFKEY, in addition we have raised issues encountered with IKE in section 4.6.4 and implementations of KINK [100] are not sufficiently mature

[12]This MTR is not used for PDNFS

made function calls into the authentication and authorization functional blocks.
We believe that the following five primitives are required for the PDNFS< − >NAT API:

- Getmappedaddress(MRSUID, data flow IP protocol type (TCP, UDP, SCTP, DCCP[11]), data flow source address, data flow source demultiplexer, data flow destination address, data flow destination demultiplexer, preferred mapped address, preferred mapped demultiplexer, portparity, contiguous-ports, NAT bind lifetime): this primitive is used to request the creation of a NAT bind, the last 5 parameters are optional. We have selected the MRSUID as a cross-reference information to avoid using new parameters. The same primitive could be used to change an existing NAT bind, for example in case the local end-host's address changes. For further information on this primitive's parameters, the reader is invited to refer to section 5.9.1.

- Returnmappedaddress(MRSUID, mappedaddress, mappeddemultiplexer, internalmapped-address, internalmapped-demultiplexer, acceptedlifetime). The internalmapped-address and internalmapped-demultiplexer are only provided when the NAT daemon implements a twice NAT function [2]

- Clearnatbind(MRSUID): this primitive is used to request an explicit NAT bind removal from the NAT bind table

- Refreshnatbind(MRSUID): this primitive is used to refresh an existing NAT bind indexed with the referenced MRSUID

- Error(MRSUID, errortype): this primitive indicates to the PDNFS layer synchronous (in response to the getmappedaddress) or asynchronous errors, the following error types and values are applicable:

  - Requested Mapped address is already used
  - Requested Mapped address and demultiplexer is already used
  - Resource exhaustion with the following values:
    * Port pool is exhausted

[11] Other protocols could be used if applicable

    * Address pool is exhausted
  - System maintenance errors:
    * NAT daemon will be shut down, NAT bind will be removed
    * Can't handle this request due to platform maintenance

7.2.1.3 PDNFS/Firewall API's Primitives Definition

We propose to use the following four primitives to allow communications between the PDNFS layer and the Firewall daemon:

- allowpf(MRSUID, data flow IP protocol type (TCP, UDP, SCTP, DCCP[11]), data flow source address, data flow source demultiplexer, data flow destination address, data flow destination demultiplexer, contiguous-ports, pinholelifetime): this primitive is used to create a pinhole for one or several flows (in case the contiguous-ports parameter is used), the last two parameters are optional (the default pinhole lifetime applicable to the transport protocol will be used by the firewall in case no lifetime was provided). We have preferred to reuse the MRSUID as the index entry for pinholes. The same primitive could be used to update an existing pinhole referenced with the same MRSUID entry.
- refresh(MRSUID): this primitive is used to refresh an existing pinhole entry
- clearpinhole(MRSUID): this primitive is used to remove an existing pinhole entry
- Error(MRSUID): this primitive notifies the PDNFS layer of synchronous (in response to allowpf) or asynchronous pinhole status changes. We propose the following error types and values:
  - Pinhole creation succeeded (error code 0)
  - Can't open pinholes for IANA assigned port numbers
  - Resource issues
    * Pinhole entry list is full
    * No more available memory for pinhole entries
    * High CPU load

- System maintenance errors:
  - Firewall daemon will be shut down, pinhole will be removed
  - Can't handle this request due to platform maintenance

7.3 Integration of PDNFS with an Application Client

We shall discuss in this section the integration of PDNFS with an application client (for example a SIP client), we believe that the proposed API primitives covered in this section could be used with any other application type. Several open source SIP implementations could be used to validate this integration model, among them the GNU osip [101] and the reSIProcate project [102] implementations.
The PDNFS< − >PDMTP API primitives defined in section 7.2.1.1 are applicable to this section as well.

7.3.1 Software Architecture Proposal

In Figure 7.2, we show the various interacting functional blocks, we shall only discuss the interaction with the authorized applications table, since the PDMTP and PDNFS functions have been described earlier in Chapters 4 and 5.

Application access to PDNFS must be restricted, as flawed applications or trojan applications could open security breaches in the network infrastructure. We propose to use an authorized application table which lists all the applications authorized to use the PDNFS. This proposed table would be owned by the root account (as discussed in section 7.1). At the time of installing an application known to have issues with NATs and Firewalls based on sections 2.4.2 and 2.4.1, the installation program would update the authorized application table; this installation operation could only be done if the user has logged in from the root account or used the sudo command and his userid was part of the system's sudoers list (as discussed in section 7.1.

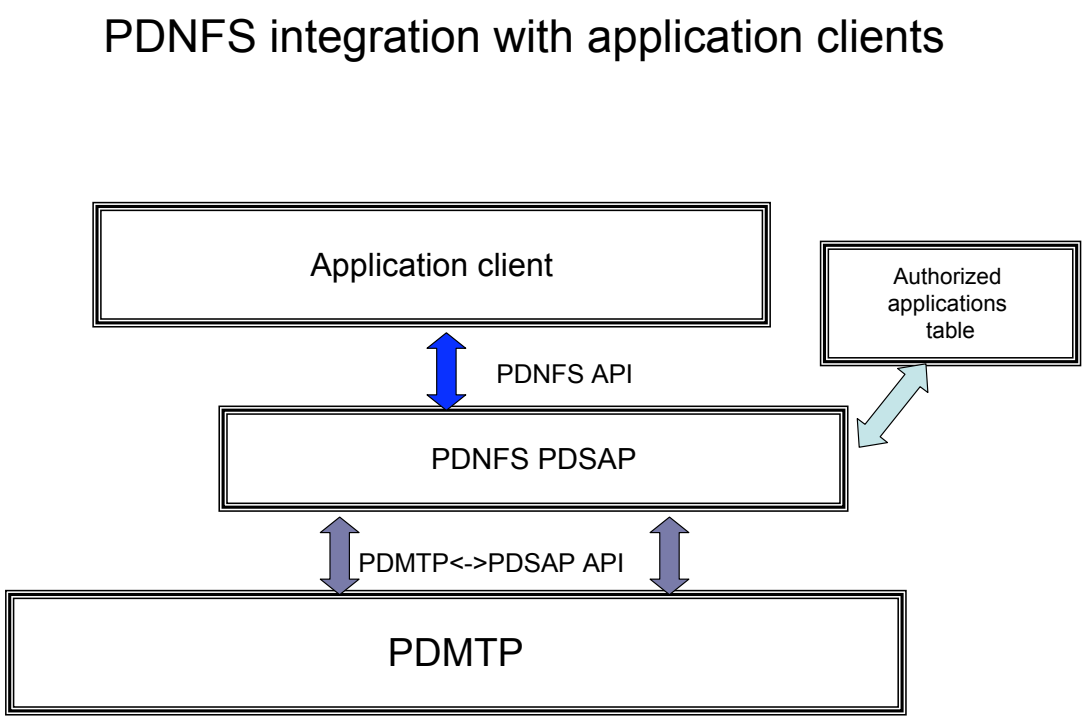


Figure 7.2: PDNFS Integration with Application Clients

7.3.1.1 PDNFS/Application Layer API's Primitive

Although the implementation example studied relates to the integration of PDNFS with a SIP client, the proposed API could be used with an application proxy.

We believe that the following six primitives are required for the Application layer$<->$PDNFS API:

- APGetmappedaddress(data flow IP protocol type (TCP, UDP, SCTP, DCCP[11]), nonce, data flow source address, data flow source demultiplexer, data flow destination address, data flow destination demultiplexer, preferred mapped address, preferred mapped demultiplexer, portparity, contiguous-ports, NAT bind lifetime): this primitive is used to request the creation of a NAT bind, the last 7 parameters are optional. The data flow destination address and destination demultiplexer might not be known at the time of using the primitive with the PDNFS layer, hence these parameters are not always pro-

vided. The same primitive could be used to change an existing NAT bind, for example in case the local end-host's address changes. For further information on this function's parameters, the reader is invited to refer to section 5.9.1.

- APReturnmappedaddress(MRSUID, list of mappedaddress(es) and mappeddemultiplexers, mappeddemultiplexer, internalmapped-address, internalmapped-demultiplexer, acceptedlifetime). The PDNFS layer uses this primitive to provide a reference index (the MRSUID) to the associated NAT and Firewall states. The internalmapped-address and internalmapped-demultiplexer are only provided when the intercepting NATs implement a twice NAT function [2]. In case several NATs were deployed within the application's host network several mapped addresses and demultiplexers would be returned (as discussed in Chapter 5).

- APallowpf(MRSUID, data flow IP protocol type (TCP, UDP, SCTP, DCCP[11]), data flow source address, data flow source demultiplexer, data flow destination address, data flow destination demultiplexer, nonce, contiguous-ports, pinholelifetime): this primitive is used by the application layer to request the creation of a pinhole for one or several flows (in case the contiguous-ports parameter is used), the last two parameters are optional. The MRSUID provided is the same one as provided by the APReturnmappedaddress primitive discussed above. The same primitive could be used to update an existing pinhole referenced with the same MRSUID entry.

- ClearMBstates(MRSUID): this primitive is used by the application layer to request from the PDNFS layer an explicit removal of all Middlebox resources (NAT bind or pinholes)

- RefreshMBstates(MRSUID): this primitive is used by the application layer to refresh all Middlebox resources associated to the referenced MRSUID

- Error(MRSUID, errortype): this primitive indicates to the application layer synchronous (in response to the APgetmappedaddress or APallowpf) or asynchronous errors, the following error types and values are applicable:

  - Pinhole creation succeeded (error code 0)

- Can't open pinholes for IANA assigned port numbers
- Requested Mapped address is already used
- Requested Mapped address and demultiplexer is already used
- User not authorized to request packet forwarding for the specific data flow descriptor
- No PDNFS aware Middlebox was found
- Resource exhaustion with the following values:
  * Port pool is exhausted
  * Address pool is exhausted
  * Pinhole entry list is full
  * No more available memory for pinhole entries
  * High CPU load
- System maintenance errors:
  * NAT daemon will be shut down, NAT bind will be removed
  * Firewall daemon will be shut down, pinhole will be removed
  * Can't handle this request due to Middlebox platform maintenance

7.4 Co-Hosted Implementation of PDNFS with the BSD PF and an Application Client

We shall discuss in this section the integration aspects of PDNFS with distributed Firewalls [103] [104].

Our work in sections 7.2 and 7.3 will be leveraged by reusing all the previously defined API primitives. Each time the application layer uses the PDNFS primitives, the PDNFS layer will use the NAT and FW APIs primitives (to request services from the local Firewall) as well as the PDNFS/PDMTP API primitives (to request services from the network's Middleboxes).

Figure 7.3 shows the required functional blocks with the associated APIs for such an implementation.

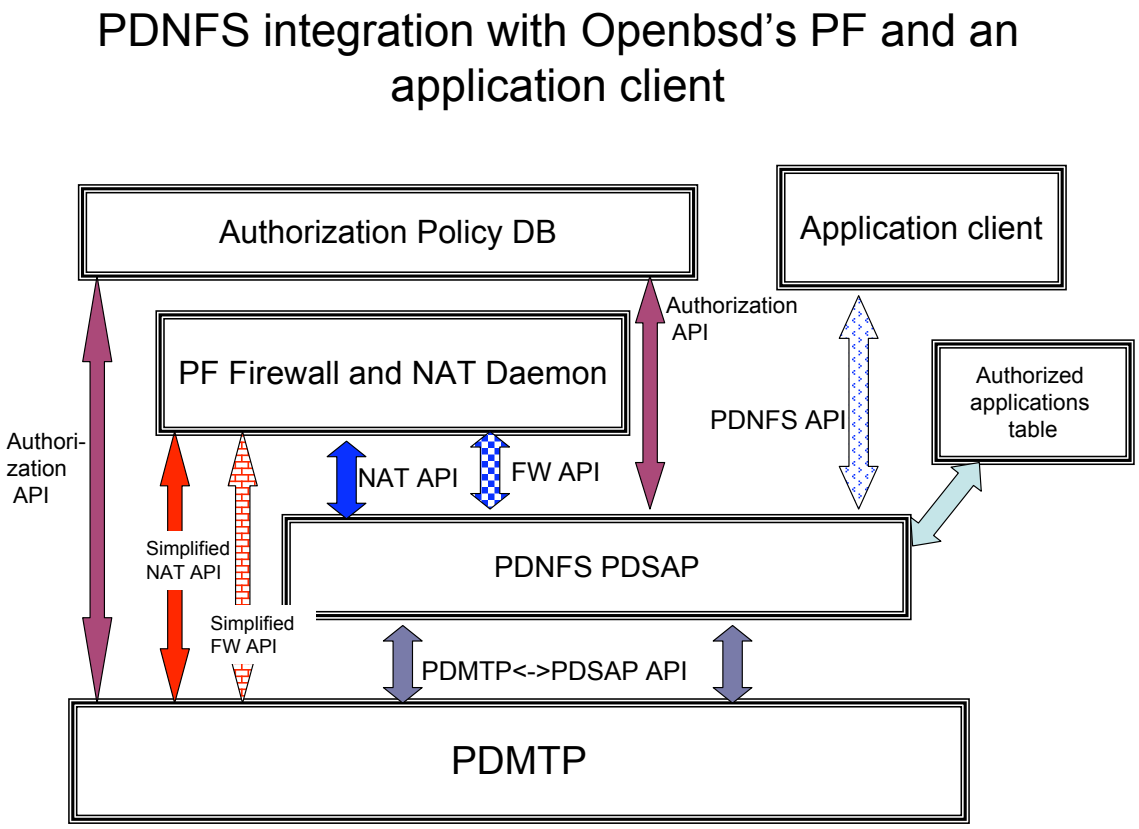


Figure 7.3: PDNFS Integration with Application Clients Co-hosting a Distributed Firewall

7.5 Summary

In this chapter we have discussed high-level implementation models of the PDNFS and PDMTP protocols on Middleboxes and application end-hosts. The proposed models are sufficiently generic to be used with any application as well as any NAT and Firewall daemons. Further refinements are expected to the proposed API primitives for global deployments of the PDNFS and PDMTP protocols.

Chapter 8

Summary and Future work

We conclude this dissertation with a summary of our work on NAT and Firewall traversal using a Path-Directed signaling framework. This will include a discussion of the merits of our NAT and Firewall signaling proposal, a short list of outstanding problems in the area of NAT and Firewall traversal where we anticipate that further study will be directed, and a summary of our main contributions.

8.1 Summary of our Path-Directed NAT and Firewall signaling Proposal

In Chapter 2, we have carefully analyzed the different types of NAT and Firewall implementations as well as a method to characterize the different impacts of NATs and Firewalls on the Internet's applications. Furthermore in that chapter we have compared the existing NAT and Firewall traversal solutions as they existed at the start of our research work.
Our comparison concluded with the selection of a NAT and Firewall signaling protocol used by application aware entities which would signal NATs and Firewalls in order to ensure the application data flow's packet traversal through arbitrary combinations of NATs and Firewalls.

In Chapter 3, we have provided a detailed study of the different types of signaling protocols by characterizing signaling protocols in two distinct families: the targeted signaling protocols and the path-directed signaling protocols.
The first protocol family has existed since the creation of the Internet, and is characterized by sending messages directly addressed to the entity that will enforce a specific behavior. The second protocol family offers a totally different paradigm that is characterized by the use of an interception mechanism for signaling messages traveling along the path which the associated

data flow is expected to take. In this path-directed paradigm, the signaling messages are addressed to the destination address (which may be the data flow's recipient address) which is a totally different address from those of the entities that are supposed to handle the signaling message and provide enforcement of the behavior signaled by the protocol. The first incarnation of the second protocol family's concept is embodied in RSVP [46]. Our comparison of targeted signaling and path-directed signaling protocol families showed that a topology insensitive NAT and Firewall signaling protocol should be a path-directed signaling protocol.

Based on this conclusion, we have analyzed the requirements for such NAT and Firewall signaling protocol and defined the overall framework for a path-directed NAT and Firewall signaling protocol.

Analysis of the framework resulted in the adoption of a two layer signaling model, where a common, lower layer is used to handle message routing and discovery of NAT and Firewall (or other types of devices with specific capabilities); the second layer comprises Path-Directed Signaling Application Protocols (**PDSAPs**) which would be specific to the capabilities being signaled (NAT binds or Firewall pinholes for NAT and Firewall signaling).

In Chapter 4, we have defined the common layer and its protocol, the Path-Directed Message Transfer Protocol (**PDMTP**) handling Middlebox discovery and message routing.

In Chapter 5, we have defined the Path-Directed NAT and Firewall Signaling (**PDNFS**) PDSAP and its use of the PDMTP protocol.

In Chapter 6, we have assessed the PDNFS protocol's requirements for the network infrastructure. In addition, we have made a detailed analysis of the protocol's real-time impacts and concluded that for real-time sensitive applications, the protocol should be used within the same administrative network to guarantee an acceptable duration for the signaling needed to enable real-time application data flow packet traversal. Furthermore, we have proposed two different security models for PDNFS deployments one where hop-by-hop security could be used (this is the simplistic approach) and an end-to-middle security model to highly restricted networks with multiple network security perimeters. The hop-by-hop security model is believed to fit in today's corporate networks using Microsoft's Active Directory [91].

An analysis covered the applicability of the PDNFS protocol to IPv6 networks where IPv4 translation will no longer be applicable.

In Chapter 7, we have proposed implementation models to integrate PDNFS with:

- an existing Firewall and NAT implementation, the OpenBSD Packet Filter (**PF**)
- an application client (a SIP client), the same model could be used for the integration with an application proxy
- an application client co-hosting a Firewall (as part of distributed firewall deployment models)

Our implementation models provided high level API primitives for all the interacting functional blocks.

8.2 NAT and Firewall Traversal Challenges for Internet Applications' Developers and Network Architects

Although IPv6 is now implemented on all the personal computers operating systems, its deployments are still marginal and limited to research institutes and engineering schools in Europe and North America. Deployments have started in Japan, China and South-Korea but most applications have not yet been enabled to support IPv6.
Application developers not only need to worry about building IP version independent software but also software that would need to interact with:

- NATs if IPv4 is used (based on the IP address selection algorithm [105])
- Firewalls

Since PDNFS has specific requirements (which cannot be met before one or two years have elapsed depending on product roadmaps) on Middleboxes (their support of PDNFS) and other parts of the network infrastructure (in case trusted third parties - KDC [57], CA [55], LDAP/RADIUS/COPS/DIAMETER servers are used for large networks), application developers should also support the proxy NAT and Firewall traversal methods (STUN or Media Proxies). The support of alternate NAT and Firewall traversal methods imposes new infrastructure components (STUN server and Media Proxies) for the transition period (and inherits

their applicability issues discussed in section 2.5.2

Our proposed work on NAT and Firewall application data flow traversal could be leveraged to allow the traversal of the application's control signaling message, however the overhead of such use is significant. A simpler approach would be to use specific pinhole creation policies (outbound traffic with specific destination transport ports) and with a keep-alive message (this was used by SIP [4] and MGCP [44]).

NAT and Firewalls hide other pitfalls for big packets exceeding MTU links on specific network segments. While solutions exist to allow the forwarding of packet fragments, these solutions could be used to launch DOS attacks on the NAT[1] or Firewalls as well as hosts (in case the attack succeeds) which have not been hardened to handle tiny fragment attacks [106] [107].

8.3 Summary of our main contributions

Our research work has provided the following:

- A method to characterize NAT and Firewall application impacts
- A comparison of existing NAT and Firewall traversal methods
- The characterization of signaling protocols into targeted signaling and path-directed signaling protocols
- A framework for path-directed signaling suitable for all network topologies, with detailed analysis of peer node discovery, message routing, stale states cleanups and refresh techniques and reuse of applications' feedback information to avoid very small periods for path-refreshes
- A Path-Directed Message Transfer Protocol suitable for multiple signaling applications enabling specific behaviors on network intermediary nodes
- A Path-Directed NAT and Firewall Signaling protocol

[1] In [5] we discussed packet fragmentation issues with all sorts of network address translators including IPv4 NATs

- Analysis of the proposed NAT and Firewall signaling protocol impacts on the network infrastructure and real-time applications
- Applicability of the proposed protocol to IPv6 deployments
- Implementation models on Firewalls, application clients/proxies hosts and application clients/proxies co-hosting a Firewall (as part of distributed Firewall deployments)

8.4 Future work

Further work is required to integrate our PDNFS proposal on application clients using multiple NAT and Firewall traversal techniques. A coordination methodology should be defined to decide which technique should be used, certain techniques could be more applicable in one network deployment than in others. The PDNFS protocol could be used in a network supporting the PDNFS protocol requirements, upon visiting a different network the host would need to use a different technique.

A NAT and Firewall state synching mechanism should be standardized while using our inputs [18] to reduce the state synching overhead.

Bibliography

[1] W.R.Cheswick, S. M. Bellovin, and A.D.Rubin. *Firewalls and Internet security: repelling the Wily Hacker*. Addison-Wesley, 1994.

[2] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT Terminology and Considerations. IETF Informational document, RFC 2663, August 1999.

[3] P.Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). IETF Informational document, RFC 3022, January 2001.

[4] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. IETF Standards Track document, RFC 3261, June 2002.

[5] C. Aoun and E. Davies. Reasons to Move NAT-PT to Experimental. IETF Draft (work in progress), draft-ietf-v6ops-natpt-to-exprmntl-02, October 2005.

[6] Packet Cable, http://www.packetcable.com.

[7] ITU-T SG16, Q.F, G, K, 2-5/16 Rapporteur Meeting, Summary of NAT and Firewall issues, AVD-2499, Beijing, 11-14 May 2004.

[8] C. Aoun, E. Davies, H. Tschofenig, and S. Thiruvengadam. Interaction of Firewalls and Network Address Translators with Internet Applications. In *IEEE Contel 2005 workshop proceedings*, June 2005.

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

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

[11] Upnp forum, Internet Gateway Device (IGD) standardized control protocol.

[12] M. Borella, J. Lo, D. Grabelsky, and G. Montenegro. Realm Specific IP: Framework. IETF experimental standard, RFC 3102, October 2001.

[13] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. SOCKS Protocol Version 5. IETF standards track document, RFC 1928, March 1996.

[14] M. Stiemerling, H. Tschofenig, and C. Aoun. NAT/Firewall NSIS Signaling Layer Protocol (NSLP). IETF draft (work in progress), draft-ietf-nsis-nslp-natfw-08, October 2005.

[15] T. Tsenov, H. Tschofenig, X. Fu, C. Aoun, and E. Davies. GIST State Machine. IETF draft (work in progress), draft-ietf-nsis-ntlp-statemachine-01, October 2005.

[16] R. Hancock, G. Karagiannis, J. Loughney, and S. van den Bosch. Next Steps in Signaling: Framework. IETF Informational document, RFC 4080, June 2005.

[17] H. Schulzrinne and R. Hancock. GIST: General Internet Signaling Transport. IETF draft (work in progress), draft-ietf-nsis-ntlp-08, October 2005.

[18] C. Aoun and O. Paul. Résolution des problèmes de routage asymétrique et de partage de charge dans les pare-feux. In *Proceedings of Colloque Francophone sur l'Ingénierie des Protocoles 2005*, April 2005.

[19] C. Aoun, E. Davies, and H. Tschofenig. Securing Middlebox Discovery for Path-Directed Signaling in the Internet. In *IEEE ASWN 2005 workshop proceedings*, July 2005.

[20] C. Aoun, E. Davies, H. Tschofenig, and M. Stiemerling. Path-directed signaling in the Internet. In *IEEE IPOM 2004 workshop proceedings*, October 2004.

[21] The OpenBSD Packet Filter, http://www.openbsd.org/faq/pf/, 2003.

[22] IANA. Special-Use IPv4 Addresses. IETF Informational document, RFC 3330, September 2002.

[23] IETF Next Steps In Signaling (NSIS) Working Group Charter.

[24] H. Tschofenig and D. Kroeselberg. Security Threats for NSIS. IETF Informational document, RFC 4081, June 2005.

[25] C. Aoun, H. Tschofenig, and M. Stiemerling. NATFW NSLP Migration Considerations. Expired IETF draft, draft-aoun-nsis-nslp-natfw-migration-02, July 2004.

[26] A. Fessi, M. Stiemerling, S. Thiruvengadam, H. Tschofenig, and C. Aoun. Security Threats for the NATFW NSLP. Expired IETF draft, draft-fessi-nsis-natfw-threats-02, July 2004.

[27] C. Aoun, H. Tschofenig, M. Stiemerling, M. Brunner, and M. Martin. NATFW NSLP Intra-Realm Considerations. Expired IETF draft, draft-aoun-nsis-nslp-natfw-intrarealm-01, July 2004.

[28] C. Aoun and N. Hamer. Potential Solutions to the Middlebox discovery problem. Expired IETF draft, draft-aoun-midcom-discovery-01, May 2002.

[29] C. Aoun. Middlebox discovery integration solutions within the Midcom architecture. Expired IETF draft, draft-aoun-middlebox-discovery-comparison-00, June 2002.

[30] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayhan. Middlebox communication architecture and framework. IETF Informational document, RFC 3303, August 2002.

[31] M. Barnes. Middlebox communications (MIDCOM) Protocol Evaluation. IETF Informational document, RFC 4097, June 2005.

[32] C. Aoun, K.Chan, L-N.Hamer, R.Penno, and S.Sen. COPS applicability as the MIDCOM PROTOCOL. Expired IETF draft, draft-aoun-midcom-cops-02, May 2002.

[33] S. Sen, C. Aoun, and T. Taylor. Applicability of MEGACO to Middlebox Control. Expired IETF draft, draft-sct-midcom-megaco-02, May 2002.

[34] Packet-based multimedia communications systems. ITU-T H.323 recommendation.

[35] F. Andreasen and B. Foster. Media Gateway Control Protocol (MGCP) Version 1.0. IETF informational document, RFC 3435, January 2003.

[36] C. Groves, M. Pantaleo, T. Anderson, and T. Taylor. Gateway Control Protocol Version 1. IETF Standards Track document, RFC 3525, June 2003.

[37] M. Holdrege and P. Srisuresh. Protocol Complications with the IP Network Address Translator. IETF Informational document, RFC 3027, January 2001.

[38] IETF MIDdlebox COMmunication (MIDCOM) Working Group Charter.

[39] B. Carpenter and S. Brim. Middleboxes: Taxonomy and Issues. IETF Informational document, RFC 3234, February 2002.

[40] E. Nordmark and R.E. Gilligan. Basic Transition Mechanisms for IPv6 Hosts and Routers. IETF draft (work in progress), draft-ietf-v6ops-mech-v2-07, March 2005.

[41] F. Audet and C. Jennings. Nat Behavioral Requirements for Unicast UDP. IETF draft (work in progress), draft-ietf-behave-nat-udp-00, January 2005.

[42] J. Rosenberg, R. Mahy, and C. Huitema. Traversal Using Relay NAT (TURN). IETF draft (work in progress), draft-rosenberg-midcom-turn-06, October 2004.

[43] C. Boulton and J. Rosenberg. Best Current Practices for NAT Traversal for SIP. IETF draft (work in progress), draft-ietf-sipping-nat-scenarios-01, October 2004.

[44] C. Aoun, M. Wakley, and T. Sassenberg. A NAT package for MGCP NAT traversal. IETF draft (work in progress), draft-aoun-mgcp-nat-package-02, February 2003.

[45] R. Braden and R. Lindell. A Two-Level Architecture for Internet Signaling. IETF draft (expired), draft-braden-2level-signaling-01.txt, November 2002.

[46] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. IETF Standards Track document, RFC 2205, September 1997.

[47] D. Katz. IP Router Alert Option. IETF Standards Track document, RFC 2113, February 1997.

[48] C. Partridge and A. Jackson. IPv6 Router Alert Option. IETF Standards Track document, RFC 2711, October 1999.

[49] Mattia Rossi and Michael Welzl. On the Impact of IP Option Processing. *Preprint-Reihe des Fachbereichs Mathematik - Informatik, No. 15*, October 2003.

[50] Mattia Rossi and Michael Welzl. On the Impact of IP Option Processing - Part 2. *Preprint-Reihe des Fachbereichs Mathematik - Informatik, No. 26*, 2004.

[51] Pierre Fransson and Andreas Jonsson. End-to-End Measurements on Performance Penalties of IPv4 Options. 2004.

[52] Load balancing with Cisco Express Forwarding. Technical report, Cisco Systems, 1998.

[53] Configuring Equal-Cost Multipath Load Sharing. Technical report, Juniper Networks.

[54] Explanation of Function: Traffic distribution across a Passport 1200 and Passport 8600 Multi-Link Trunk (MLT), Customer Support Bulletin. Technical report, Nortel, 2001.

[55] Information technology Open systems interconnection The Directory: Public-key and attribute certificate frameworks. ITU-T X.509 standard, 2000.

[56] R. Moskowitz and P. Nikander. Host Identity Protocol Architecture. IETF draft (work in progress), draft-ietf-hip-arch-02, January 2004.

[57] C. Neuman, T. Yu ans S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). IETF Standards Track document, RFC 4120, July 2005.

[58] L-N. Hamer, B. Gage, and H. Shieh. Framework for session set-up with media authorization. IETF Standards Track document, RFC 3521, April 2003.

[59] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. IETF Standards Track document, RFC 3550, February 2005.

[60] Data protocols for multimedia conferencing. ITU-T T.120 recommendation.

[61] FreeBSD UNIX, http://www.freebsd.org.

[62] R. Braden. Requirements for internet hosts – communication layers. IETF Standards Track document, RFC 1122, October 1989.

[63] Sun Solaris UNIX Operating System, http://www.sun.com/software/solaris/index.jsp.

[64] IBM AIX UNIX Operating System, http://www-03.ibm.com/servers/aix/os/53desc.html.

[65] TCG Specification Architecture Overview. TCG document, April 2004.

[66] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. IETF Standards Track document, RFC 2474, December 1998.

[67] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification). IETF Standards Track document, RFC 2460, December 1998.

[68] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. IETF Standards Track, RFC 2401, November 1998.

[69] T. Dierks and C. Allen. The TLS Protocol Version 1.0. IETF Standards Track, RFC 2246, January 1999.

[70] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address. IETF Best Current Practices document, RFC 2827, May 2000.

[71] F. Baker and P. Savola. Ingress Filtering for Multihomed Networks. IETF Best Current Practices document, RFC 3704, March 2004.

[72] Internet Assigned Number Authority. Ipv6 router alert option values.

[73] R. Rosenberg. Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Multimedia Session Establishment Protocols. IETF draft (work in progress), draft-ietf-mmusic-ice-03, October 2004.

[74] G. Camarillo and J. Rosenberg. The Alternative Network Address Types (ANAT) Semantics for the Session Description Protocol (SDP) Grouping Framework. IETF Standards Track document, RFC 4091, June 2005.

[75] T. Kivinen, B. Swander, A. Huttunen, and V. Volpe. Negotiation of NAT-Traversal in the IKE. IETF Standards Track document, RFC 3947, January 2005.

[76] A. Huttunen, B. Swander, V. Volpe, L. DiBurro, and M. Stenberg. UDP Encapsulation of IPsec ESP Packets. IETF Standards Track document, RFC 3948, January 2005.

[77] C. Huitema. Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP). IETF Standards Track document, RFC 3605, October 2003.

[78] C. Aoun, H. Tschofenig, and M. Stiemerling. NAT/Firewall NSLP Intra-Realm Considerations. Expired IETF draft, draft-aoun-nsis-nslp-natfw-intrarealm-01, July 2004.

[79] H. Schulzrinne, A. Rao, and R. Lanphier. REAL Time Streaming Protocol. IETF Standards Track document, RFC 2326, April 1998.

[80] J. Hodges and R. Morgan. Lightweight Directory Access Protocol (v3): Technical Specification. IETF Standards Track document, RFC 3377, September 2002.

[81] R. Housley, T. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. IETF Standard Track document, RFC 3280, April 2002.

[82] R. Housley. Cryptographic Message Syntax (CMS). IETF Standards Track document, RFC 3852, July 2004.

[83] R. Housley. Protecting multiple contents with the Cryptographic Message Syntax (CMS). IETF Standards Track document, RFC 4073, May 2005.

[84] Open SSL Project, http://www.openssl.org.

[85] Common Address Redundancy Protocol, Open BSD man pages, http://www.openbsd.org/cgi-bin/man.cgi?query=carp, 2003.

[86] K. Seo C. Lynn, S. Kent. X.509 Extensions for IP Addresses and AS Identifiers. IETF Standard Track document, RFC 3779, June 2004.

[87] A. Gulbrandsen, P. Vixie, and L. Esibov. A DNS RR for specifying the location of services (DNS SRV). IETF Standards Track document, RFC 2782, February 2000.

[88] B. Tung and L. Zhu. Public Key Cryptography for Initial Authentication in Kerberos. IETF Draft (work in progress), draft-ietf-cat-kerberos-pk-init-29, October 2005.

[89] G. Tsirtsis and P. Srisuresh. Network Address Translation - Protocol Translation (NAT-PT). IETF Standards Track document, RFC 2766, February 2000.

[90] G. Van de Velde, T. Hain, R. Droms, B. Carpenter, and E. Klein. IPv6 Network Architecture Protection. IETF Draft (work in progress), draft-ietf-v6ops-nap-02, October 2005.

[91] Windows 2003 Active Directory, http://www.microsoft.com/windowsserver2003/technologies/direct

[92] Portable Operating System Interface (POSIX). IEEE 1003 specification.

[93] Max Laier. Packet Filter (pf), An Extended Introduction, http://people.freebsd.org/mlaier, 2004.

[94] PF : The OpenBSD Packet Filter - FAQ, http://www.openbsd.org/faq/pf/index.html, 2005.

[95] J. Linn. Generic Security Service Application Program Interface Version 2, Update 1. IETF Standards Track document, RFC 2743, January 2000.

[96] L. Zhu, K. Jaganathan, and S. Hartman. The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2. IETF Standards Track document, RFC 4121, July 2005.

[97] A. Medvinsky and M. Hur. Addition of Kerberos Cipher Suites to Transport Layer Security (TLS). IETF Standards Track document, RFC 2712, October 1999.

[98] P. Eronen and H. Tschofenig. Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). IETF draft (work in progress), draft-ietf-tls-psk-09, June 2005.

[99] W.R. Stevens and B. Fennerand A.M. Rudoff. *UNIX Network Programming: The Sockets Networking API. Volume 1 Third Edition.* Addison-Wesley, 2004.

[100] S. Sakane, K. Kamada, M. Thomas, and J. Vilhuber. Kerberized Internet Negotiation of Keys (KINK). IETF draft (work in progress), draft-ietf-kink-kink-07, May 2005.

[101] The GNU oSIP library, http://www.gnu.org/software/osip/osip.html.

[102] The sip Foundry ReSIProcate project, http://www.sipfoundry.org/resiprocate/.

[103] S.M. Bellovin. Distributed firewalls. In *Login*, pages 37–39, November 1999.

[104] S. Ioannidis, A.D. A.D. Keromytis, S.M. Bellovin, and J.M. Smith. Implementing a distributed firewall. In *Proceedings of Computer and Communications Security (CCS), Athens, Greece*, pages 190–199, November 2000.

[105] R. Draves. Default Address Selection for Internet Protocol version 6 (IPv6). IETF Standards Track document, RFC 3484, February 2003.

[106] G. Ziemba, D. Reed, and P. Traina. Security Considerations for IP Fragment Filtering. IETF Informational document, RFC 1858, October 1995.

[107] I. Miller. Protection Against a Variant of the Tiny Fragment Attack. IETF Informational document, RFC 3128, June 2001.

Index