Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: March 21, 2011

J. Ubillos
Swedish Institute of Computer Science
M. Xu
Z. Ming
Tsinghua University
C. Vogt
Ericsson

September 17, 2010

Name-Based Sockets Architecture
draft-ubillos-name-based-sockets-03

Abstract

This memo defines Name-based sockets. name-based sockets allow application developers to refer to remote hosts (and it self) by name only, passing on all IP (locator) management to the operating system. Applications are thus relieved of re-implementing features such as multi-homing, mobility, NAT traversal, IPv6/IPv4 interoperability and address management in general. The operating system can in turn re-use the same solutions for a whole set of guest applications. name-based sockets aims to provide a whole set of features without adding new indirection layers, new delays or other dependences while maintaining transparent backwards compatibility.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 21, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Conventions 3
2. Terminology 3
3. Overview 3
  3.1. Introduction 3
  3.2. Motivation 3
  3.3. Compatibility goals 4
    3.3.1. Network compatibility 4
    3.3.2. Application compatibility 4
  3.4. name-based sockets overview 4
  3.5. Protocol overview 5
4. Initial name exchange 5
  4.1. Name format 7
  4.2. Service names (and port numbers) 7
  4.3. Transport selection 7
5. API 7
6. Security Considerations 9
7. IANA Considerations 9
8. Contributors 9
9. References 9
  9.1. Normative References 9
  9.2. Informative References 9
  9.3. URL References 9

1. Conventions

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

2. Terminology

Locator - An IP address (v4 or v6) on which a host can be reached.

Name - A character string (max 255 chars long) on which a host can be identified.

3. Overview

3.1. Introduction

name-based sockets provide a unified interface which caters to the application developers who wish to simply open up a communication to a remote host by its name and have the operating system perform the management of locators.

It does so by providing a new address family (AF_NAME) which allows the developer to use a name instead of an IP (or other locator).

3.2. Motivation

Network communication has for very long been based on the assumption that applications should deal with the IP (locator) management. This is based on the legacy notion that an IP does not change during a session and that a session is a communication between two given IPs (locators). This situation has changed, locators change during a session, and a device/host might have multiple locators, hosts may be behind NA(P)Ts or be on different locator domains (e.g. IPv4/IPv6). Today, this is mostly left to the individual applications to solve. This adds complexity to the applications making it a challenge in it

self just to meet the networking demands of applications.

Name-based sockets aims to fix this by pushing the locator management to the operating system, without introducing any new limitations or delays.

By approaching the problem at the socket() level, existing abstraction frameworks can easily benefit from a change. By allowing the existing abstractions to simply pass along the name the whole way down to the socket() call, the abstraction or whole framework are able to benefit by name-based sockets.

Note that name-based sockets do not aim to replace all socket() based communications. There are of course cases which are limited due to obvious boot-strapping problems. E.g. a DHCP client, or an DNS-querying client would do better in not using a name oriented architecture.

3.3. Compatibility goals

3.3.1. Network compatibility

1. Minimal changes to "on the wire" protocol.

2. No changes to added 'features'

3.3.2. Application compatibility

The objective is to maintain the (BSD) socket() semantics commonly used today. It is important that the transition for developers should be trivial.

Maintaining these semantics allows existing abstraction frameworks to integrate name-based sockets to their frameworks. Socket abstractions are often used to simplify the use of a service e.g. HTTP. In those cases, they do not add functionality at the network or transport layer. These kinds of abstraction frameworks would quickly be able to make use of the extra functionality provided by name-based sockets.

3.4. name-based sockets overview

name-based sockets provide a new socket interface. It is implemented using a new address family (AF_NAME). This means that the sockets are only used by applications who explicitly invoke it. The result is that applications that do use name based sockets are very aware of the set of features they are provided with, hence they know not to re-implement them. Current implementations are implemented as kernel modules, however a user-space library implementation ought to work just as well.

By using DNS as the ID/Locator mapping structure, we do not introduce any new indirections. Please note that the responding host does not need to do any reverse-DNS resolution (explained below). We part from the assumption that most application network calls start with a FQDN.

The exchange of names is performed in-band, by piggy-backing the needed information on the first couple of packets exchanged. The consequence is:

- o No extra delays
- o No extra features until the name exchange has ended.
- o As a result of this, we also achieve backwards compatibility (a name exchange which never completes)

3.5. Protocol overview

name-based sockets work by performing a name exchange in the beginning of a communication. It does this in an non-blocking way. In practice what happens is that the first few packets are exchanged in a normal legacy fashion. However, to these packets, extra information about the corresponding hosts names are piggy-backed. If a name exchange is successful, the extra features provided by name-based sockets are enabled. If the exchange does not succeed, normal legacy communication continues unaffected.

The name of each host is either its FQDN, its IP in IPv6.arpa format or an arbitrary nonce. It may or may not be authenticated. In the ordinary case, the name is not authenticated, thus the receiver does not need to perform a reverse or forward lookup, hence not adding any

further delays to the first packet(s). The motivation for this is to avoid any additional "first-packet" delays.

Once the name exchange has been performed successfully the complete feature set will be made available to the communication automatically.

The expected API for a socket using AF_NAME is the same as for e.g. TCP (SOCK_STREAM). This is also the case for SOCK_DGRAM and similar protocols, for all practical purposes, the functionality remains unchanged, however, as state is created in both ends, connection oriented semantics are more intuitive.

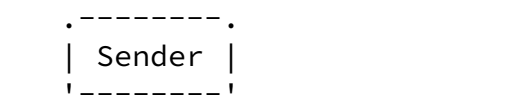
4. Initial name exchange

When the sender sends the first packet to the receiver it appends its own name as an IP-Option/IPv6-Extension header. It repeats this for a predefined amount of time or packets. On the receiving end, if the receiver supports name-based sockets it appends its own name in the same fashion for a predefined amount of time or packets. Should the receiver not be able to interpret the name, the option/extension header is ignored and the legacy communication precedes as normal.

This kind of name exchange has two consequences. First and foremost that there are no extra delays on the initial packets. Secondly that the complete feature set provided by name based sockets will not be

available until a few packets have been exchanged.

In figure 1 the exchange is depicted. The first packet from the sender has its own name appended to it. The next few packets sent will also have the name appended to it for a predefined number of packets (X in the figure) or until a reply including a name extension is received. On the receiving end, should an incoming packet have a name extension, the receiver begins to append its own name to the sent packets. It does so for a predefined number of packets (X in the figure).



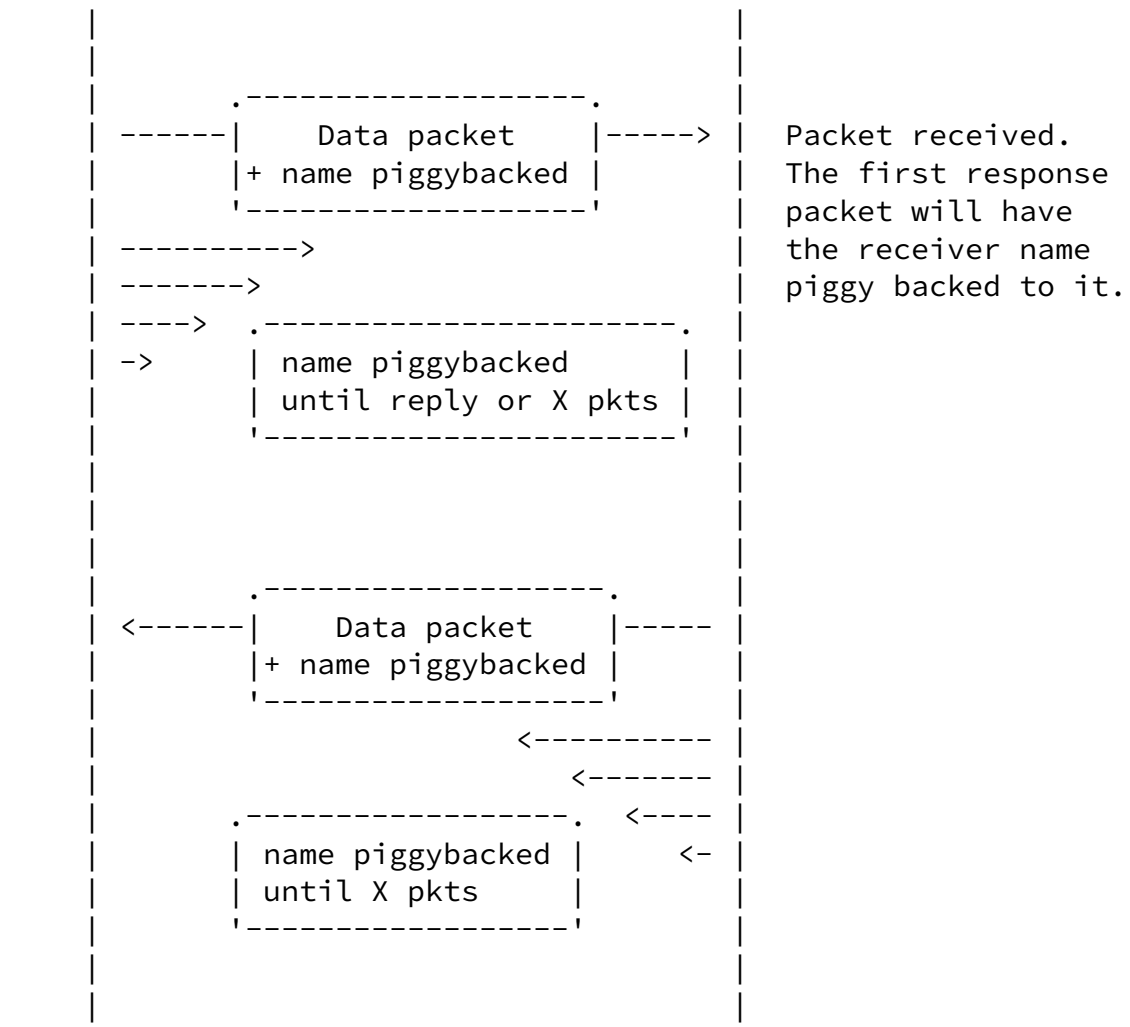


Figure 1

4.1. Name format

Names can be provided in any of three ways.

- FQDN. The Fully Qualified Domain Name of the host. This will allow e.g. DNSsec to provide authenticity of the name.

- ip6.arpa. Using one of the hosts interfaces addresses as a name.

o Nonce. A one-use only session identifier.

4.2. Service names (and port numbers)

Services are identified by a string. For compatibility reasons, it is reasonable to use the "Keyword" field as an identifier of the service in question.

Excerpt from http://www.iana.org/assignments/port-numbers

Keyword	Decimal	Description	References
http	80/tcp	World Wide Web HTTP	
http	80/udp	World Wide Web HTTP	
www	80/tcp	World Wide Web HTTP	
www	80/udp	World Wide Web HTTP	
www-http	80/tcp	World Wide Web HTTP	
www-http	80/udp	World Wide Web HTTP	
http	80/sctp	HTTP	

Keyword and port number mappings.

4.3. Transport selection

Often today, the transport protocol is implied by the service in question. How this is handled remains to be defined. It is however likely that a default transport protocol can be provided depending on which service is requested.

5. API

The API follows the connection oriented model, e.g. TCP. The available calls are:

listen(): Prep for incoming session


```
fd = listen( local_name, peer_name, service, transport );
```

open(): Initiate outgoing session

```
fd = open( local_name, peer_name, service, transport );
```

accept(): Receive incoming session

```
accept( peer_name, fd );
```

read(): Receive data

```
data = read( fd );
```

write(): Send data

```
write( fd, data );
```

close(): Close session

```
close( fd );
```

Note: In the above examples

local_name The local identifier. Either an explicit name or a wildcard (*). As host may have multiple names, to choose which to listen to, a name must be chosen or a wildcard may be used. In the latter case, on listening, any destination name is accepted; on send, an arbitrary name (valid for the host) may automatically be chosen by the socket.

peer_name The remote identifier. Either an explicit name or a wildcard (*). In the accept() function, a wildcard might be inserted. In this case, all incoming packets will be accepted, independent of sender name.

service The service to be run. In general this will correspond to the keyword field in the IANA Port Numbers registry. E.g. http, ftp and ssh.

transport Transport refers to the chosen transport protocol. This could be e.g.

* TCP (SOCK_STREAM)

* UDP (SOCK_DGRAM)

* SCTP (SOCK_SCTP)
* DCCP (SOCK_DCCP)
* NORM ...
* And so on ...

6. Security Considerations

7. IANA Considerations

name-based sockets requires a new address family (AF_NAME) to be defined.

8. Contributors

9. References

9.1. Normative References

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

9.2. Informative References

[RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, June 2009.

[RFC5534] Arkko, J. and I. van Beijnum, "Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming", RFC 5534, June 2009.

[I-D.cotton-tsvwg-iana-ports] Cotton, M., Eggert, L., Mankin, A., and M. Westerlund, "IANA Allocation Guidelines for TCP and UDP Port Numbers", draft-cotton-tsvwg-iana-ports-00 (work in progress), February 2008.

9.3. URL References

Authors' Addresses

Javier Ubillos
Swedish Institute of Computer Science
Kistagangen 16
Kista
Sweden

Phone: +46767647588
EMail: jav@sics.se

Mingwei Xu
Tsinghua University
FIT Building 4-104, Tsinghua University
Beijing
China

EMail: xmw@cernet.edu.cn

Zhongxing Ming
Tsinghua University
FIT Building 4-104, Tsinghua University
Beijing
China

EMail: mingzx@126.com

Christian Vogt
Ericsson
200 Holger Way
San Jose, CA 95134-1300
USA

EMail: christian.vogt@ericsson.com