A TCP Option for Port Names
draft-touch-tcp-portnames-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on October 14, 2006.

Abstract

This document specifies a new TCP option that specifies services by a string name. This option separates the use of port numbers for connection demultiplexing from their use as a service identifier. The option allows a larger number of concurrent connections for a particular service, as well as potentially enabling more explicitly coordination of services behind NATs and firewalls. This option should be supported in new TCP implementations.
Conventions used in this document

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]

">>" precedes all RFC-2119 recommendations, which are listed separately at the end of Section 1: Introduction.

Table of Contents

1. Introduction..................................................2
2. Motivation....................................................3
   2.1. IANA Port Numbers........................................4
   2.2. DNS SRV Records..........................................5
   2.3. RPC Portmapper and RPCBIND..............................5
   2.4. TCPMUX..................................................6
   2.5. Summary and Proposal....................................6
3. The TCP Portname Option......................................8
   3.1. Interactions Between Portnames and Port Numbers.........9
   3.2. Error Conditions........................................10
   3.3. Backward Compatibility..................................10
4. Issues........................................................11
   4.1. API Support.............................................11
   4.2. Interaction with Other Protocols........................11
   4.3. Potential Use in Other Transport Protocols...............13
   4.4. Discussion of Alternative Approaches....................14
5. The Portname Name Space....................................14
6. Security Considerations....................................16
7. IANA Considerations.........................................17
8. Conclusions..................................................17
9. Acknowledgments.............................................17
10. References..................................................18
    10.1. Normative References................................18
    10.2. Informative References.................................18
Author’s Addresses............................................20
Intellectual Property Statement................................20
Disclaimer of Validity..........................................21
Copyright Statement............................................21
Acknowledgment................................................21

1. Introduction

Most Internet transport protocols use "well-known" port numbers to indicate which application service is associated with a connection or message; this includes TCP, UDP, SCTP, and DCCP [RFC768] [RFC793]
Making a port number well-known involves registration with the Internet Assigned Numbers Authority (IANA), which includes defining a service by a unique keyword and reserving a port number from among a fixed pool [IANA].

This document specifies the TCP portname option, which allows services to be specified by a string. This decouples the use of ports for connection demultiplexing and state management from their use to indicate a desired endpoint service. It also conserves port numbers by allowing endpoints to allocate their own port numbers separately, based on services they deploy.

This option allows a flexible correspondence between services and port numbers, which affects how applications interact with TCP, as well as how services can be deployed behind NATs and firewalls. Although it changes certain TCP segments (SYNs), it does not otherwise affect the processing of TCP segments or the TCP state machine. The option must be implemented at both ends of a TCP connection for benefit. The option affects only the initiator of a TCP connection, and fails as if the service were not available when not supported at the receiver.

The following is a summary of the RFC2119-level recommendations of this document:

>> (TO BE COMPLETED PRIOR TO FINAL SUBMISSION)

2. Motivation

TCP supports multiplexing as one of its six core services, allowing a single pair of hosts to have multiple concurrent TCP sessions [RFC793]. An endpoint address is associated with a port number, forming a socket; and "A pair of sockets uniquely identifies each connection." Although ports can be bound to services uniquely at each endpoint, RFC 793 notes that it is useful to attach frequently-used services to fixed ports which are publicly known, but that other services may be discovered by dynamic means. This document considers the impact of that suggestion, and specifies an alternative which achieves similar coordination.

The Internet currently relies on the use of fixed, publicly-known ports for most services, whether intended for public access (e.g., HTTP, FTP, DNS) or for services typically used between pre-arranged pairs (e.g., X11, SSL). Some of these services use one public port to negotiate other ports for further exchanges (e.g., FTP, H.323, RPC).
There are three current methods for determining the port for a public service:

- Index the service in IANA’s port registry
- Index the service in the host’s DNS SRV records
- Ask the host directly using an RPC portmapper/bind-like service
- Ask the host for a hand-off using the TCPEMUX port (port #1)

Many of these alternatives, including the use of strings as service identifiers, were described in principle in RFC 814, and have evolved into deployed capabilities [RFC814]. Each of these alternatives is summarized below, and each either limits the number of concurrent connections for a service unnecessarily or incurs additional latency or management overhead compared to the portname option presented in Section 3.

2.1. IANA Port Numbers

The Internet Assigned Numbers Authority currently manages globally reserved port numbers [IANA]. The desired port number for a service is determined either by an operating system index to a copy of IANA’s table (e.g., getportbyname() in Unix, which indexes the /etc/services file), or is fixed in inside the application.

The port number space - 0..65536 - is split into three ranges [RFC2780]:

- 0..1023 "well-known", also called "system" ports
- 1024..49151 "registered", also called "user" ports
- 49152..65535 "dynamic", also called "private" ports

The terms "well-known" and "registered" are misnomers; both of those port ranges are managed by IANA, and are equally registered and well-known. System ports are intended for services that run in privileged mode, sometimes known as "root", although that distinction is blurred in current operating systems.

The larger challenge with IANA-managed ports is that it allocates ports globally, for all hosts everywhere on the public Internet, even though the meaning of a port need be known only for a particular host. As a result, the fixed space of port numbers is being globally reserved unnecessarily. It would be more useful to allocate this name
space on a per-host basis. It is optional whether such a name space would retain any of the system/user/dynamic distinctions of the current port number space.

2.2. DNS SRV Records

DNS SRV resource records provide a way to find the port number for a service based on its string name [RFC2782]. A host asks the DNS to index "_portname_.hostname" (underscores required) and the response is a record that includes both the port number and host’s IP address.

SRV records allow port numbers to be allocated on a per-host basis. This lookup requires an additional protocol exchange for each first-time access, which can traverse much of the same path the TCP SYN will, i.e., incurring an additional round-trip time of delay (because DNS servers are often located near the hosts they serve).

The larger challenges for DNS SRV records are autonomy, robustness, and size of the name space. Many hosts do not have control over their DNS entries; moving port lookup into the DNS could limit the services that a host can deploy for public access. This solution also makes the DNS a required part of the Internet architecture, even for accessing services on hosts with well-known IP addresses (e.g., the DNS itself). This decreases network robustness, because access of services on a host depends on access to the DNS. A more efficient, fate-sharing approach is desired. Finally, DNS SRV records return a conventional port, limiting the number of available services on a DNS name to 65,536.

2.3. RPC Portmapper and RPCBIND

An alternative to indexing the portname at a separate host via the DNS would be to contact the intended host directly and request the lookup there. This is how the RPC portmapper (v2) and RPCBIND (v3 and v4) services work, where the source host contacts the destination on port #111 [RFC1831][RFC1833]. This service was designed for the same basic reason as the TCP portname option of this document: to allow a small subset of a potentially large set of services to be dynamically bound to a small number of ports. The differences between portmapper and RPCBIND are not important here, so they are discussed as a single example.

In both portmapper and RPCBIND the source host contacts the destination host on port 111, and issues a request including the desired destination RPC service name. A response indicates the appropriate port for that RPC service.
Like the DNS SRV solution, portmapper/RPCBIND requires a separate round-trip (one for UDP; more for TCP) to perform the lookup operation. This adds to both the communication overhead and connection establishment latency.

The portmapper service also allows services to be selected on version, i.e., to have different versions of a service on different ports, accessed using the same version name but a different version number. This is handled in IANA, DNS SRV records, and TCPMUX by using a port keyword that embeds the version number in the name, e.g., "imap" vs. "imap3"; the remainder of this document considers this a sufficient solution to versioning.

2.4. TCPMUX

TCPMUX is a service on TCP port #1 which allows a host to provide a port name handoff service for itself [RFC1078]. A source host opens a connection to port 1 on a destination host and transmits "portname<CR><LF>" in the data stream; the destination replies with either "+<CR><LF>" (yes, the service is available) or "-<CR><LF>" (no, the service is not available). If the service is available, the connection is transferred to the desired service while still in the OPEN state.

TCPMUX modifies the semantics of TCP connection establishment; its connections always succeed, and upon receipt of the named service the application must determine whether to proceed or not. This document seeks a more conventional TCP semantics, where unavailable services result in a rejected connection (e.g., RST in reply and/or ICMP error message).

TCPMUX further requires all new connections to be received on a single port; this limits the number of connections between two machines to 65,536; while this is a reasonably high number, it may not be appropriate for services with connection-per-transaction semantics.

2.5. Summary and Proposal

Each of the alternatives presented has a significant limitation. These limits are summarized as follows:

- IANA ports: limited to 65,536 services throughout the Internet; fewer if system/user/dynamic boundaries are preserved
DNS SRV records: requires an extra round-trip exchange for lookup, not typically under host control, limited to 65,536 services per DNS name (fewer if system/user/dynamic boundaries are preserved)

Portmapper: requires an extra round-trip exchange for lookup

TCPMUX: destroys semantics of TCP connection establishment, limits services and connections per endpoint pair to 65,536

The TCP portname option allows the destination host to associate named services with processes on a per-connection basis, while avoiding unnecessary additional round-trips or connections and while also reducing message overhead.

The basic operation of the portname option is as follows:

- The source host issues a SYN, picking both source and destination port numbers randomly that are not currently in-use to that destination.
- The SYN includes the portname option, which contains the string name of the desired service.
- The destination host, upon receiving the SYN with the portname option, determines whether an available service matching the string is running which admits dynamically-bound port pairs. If so, a SYN-ACK is issued with a zero-length portname option, indicating success of the lookup and handoff. The service is bound to that connection at the destination.
- If the service is not available, the appropriate RST and/or ICMP error messages are returned.

The benefits to the TCP portname option are that:

- The number of services available is no longer limited by 65,536; the limit is based solely on the number of unique connections between two hosts (i.e., 4,294,967,296)
- Portname support is provided at the same host as the intended service, so the fate of both is shared (more robust)
- The option is embedded in the TCP SYN segment, avoiding extra round trips and messages.
- TCP connection semantics are maintained, i.e., services not available never connect.
3. The TCP Portname Option

The TCP portname option extends the TCP header to include a string indicating the name of a service, as shown in Figure 1.

>> New implementations of TCP SHOULD implement the portname option.

>> The TCP portname option MAY appear in any TCP segment, but it SHOULD NOT be added any segments except SYNs and SYN-ACKs.

>> The option MUST be ignored if in any segments except SYNs and SYN-ACKs.

The option includes the mandatory KIND and LENGTH fields, as well as the string name. The KIND is a single octet (byte) which indicates that this option is the TCP portname option. The LENGTH is a single octet (byte) interpreted as an unsigned number that indicates the length of this option in octets (bytes), including the KIND and LENGTH fields, as well as the octets of the name string.

>> The LENGTH field MUST be greater than or equal to 2.

The NAMESTRING field is a variable-length UTF-8 string that represents the name of a service; for most typical services, this is equivalent to a US ASCII string.

>> The NAMESTRING MAY be of any length that fits in the available TCP header space, including zero.

+--------+--------+------...
|  KIND  | LENGTH |  NAMESTRING
+--------+--------+------...
   Kind   Length

Figure 1 TCP portname option format

Upon receipt of a TCP SYN segment including the portname option ("TCP SYN/portname"), the NAMESTRING is matched against a list of available services.

>> The NAMESTRING MUST match the service name exactly to consider the indexing valid.

The way in which the portname option and TCP destination port numbers interacts is described in Section 3.1. When an incoming TCP SYN/portname is considered valid, the connection is completed by returning a SYN-ACK with the TCP portname option.
A TCP SYN-ACK in response to a TCP SYN/portname MUST include a null portname option, i.e., with LENGTH=2.

A TCP SYN/portname answered with a TCP SYN with a non-null portname option (LENGTH > 2) or lacking the portname option MUST cause the initiator to abort the connection via issuing a RST and by reporting an error to the application as if the port were not available.

The TCB for that connection is then associated with the process for the matching service, which then handles all further interactions with the connection.

3.1. Interactions Between Portnames and Port Numbers

TCP currently uses TCP port numbers to demultiplex connections as well as to indicate the desired service at the destination. The portname option retains the demultiplexing capability, but overrides service identification.

TCP specifies port numbers in the OPEN command, which corresponds to Unix bind() and connect() system calls. The current OPEN command is described in RFC 793 Sections 2.7 and 3.8 as:

```
OPEN (local port, foreign socket, active/passive
 [, timeout] [, precedence] [, security/compartment]
 [, options])
 -> local connection name
```

The portname option requires the TCP OPEN call to be augmented so that the local port and foreign socket parameters include portnames as well as port numbers, e.g., "portname.portnumber".

In specific, this modifies RFC 793 to refer to "port indicator" rather than local port or remote port, where a port indicator would be the pair "portname.portnumber". In such a port indicator, either the portname or portnumber could be indeterminate (e.g., "DNS.*", or "*.80". E.g., an incoming connection to port 28 with a portname of "DNS" would be interpreted as "DNS.28".

Existing application bind requests to port numbers, in the absence of the portname option, MUST be interpreted as if the keyword for that port number were prepended to the port number.

E.g., an application binding to port 80, using no portname option, would be equivalent to "HTTP.80". For ports lacking existing keywords, the portname is substituted with "UNKNOWN".
An implementation MUST allow applications to bind to a portname on a fixed port.

E.g., an application could bind to port 80 and indicate portname "DNS" to bind to "DNS.80".

An implementation MUST allow applications to bind to a portname without specifying a fixed port.

E.g., an application could bind to port INPORT_ANY and indicate portname "DNS" to bind to "DNS.INPORT_ANY".

As with INADDR_ANY, we anticipate that INPORT_ANY will be overridden by more specific port identifiers.

3.2. Error Conditions

There are a number of error conditions for a SYN segment with the portname option to be considered:

- Invalid NAMESTRING
- Invalid service
- Invalid port

In all three cases, the receiving TCP should act as it would if the service were not available, i.e., it SHOULD return an ICMP port unreachable error message [RFC1122]. This message MUST include the received TCP header including the portname option in its entirety. The destination TCP MUST also send a RST in response. Other interactions are the result of backward compatibility, and are discussed in Section 3.3.

3.3. Backward Compatibility

The TCP portname option is designed to interact correctly only with portname-supporting implementations, and will fail to connect with non-portname implementations. Applications intended to be compatible with both implementations MUST either attempt both portname and non-portname connections in parallel or retry a failed portname attempt with a non-portname attempt.

There are two error conditions due to the interaction of TCPs supporting the portname option with TCPs which lack that support. In particular:
o As per existing RFC793 requirements TCP SYN segments received by a non-portname TCP MUST be ignored, and will return a SYN-ACK if any service bound to that port. The source TCP will thus receive a SYN-ACK without the required null portname option (i.e., LENGTH=2), which will cause the connection attempt to be aborted via a RST and an error sent to the application as if the port were unreachable.

o Conventional TCP SYN received by portname TCP: A portname-capable TCP MUST allow conventional binding of services to fixed ports. TCP SYNs lacking the portname option MUST be handled conventionally.

4. Issues

The TCP portname option requires API support, one variant of which is informally presented here. The option interacts with some other IP and TCP services, notably security services. Variants of the option may be useful in other transport protocols. Also, there were a number of alternate designs considered which this document captures in summary.

4.1. API Support

The TCP portname option requires an API to allow a service to bind to a port NAMESTRING rather than just a port number. One approach would be to extend the existing port number fields to allow the "portname.portnumber" format. A simpler approach is to use separate commands as follows:

Use the existing port number indicator command (e.g., Unix bind() or connect() calls) to select a specific port number where desired.

Use the existing socket parameterization command (e.g., Unix setsockopt()) to set the portname option, e.g., TCP_PORTNAME).

We propose to extend the semantics of ‘all zeroes’ as used to indicate floating/dynamically selected IP addresses and port numbers to be used for the unspecified port number INPORT_ANY.

4.2. Interaction with Other Protocols

The TCP portname option potentially interacts with any other protocol that interprets or modifies TCP port numbers. This includes IPsec and other firewall systems, TCP/MD5 and other TCP security mechanisms, FTP and other in-band exchange of ports, and network address translators (NATs).
IPsec uses port numbers to perform access control in transport mode [RFC4301]. Security policies can define port-specific access control (PROTECT, BYPASS, DISCARD), as well as port-specific algorithms and keys. Similarly, firewall policies allow or block traffic based on port numbers.

Use of port numbers in IPsec selectors and firewalls may assume that the numbers correspond to well-known services. It is useful to note that there is no such requirement; any service may run on any port, subject to mutual agreement between the endpoint hosts. Use of the portname option may interfere with this assumption both within IPsec and in other firewalling systems, but it does not add a new vulnerability. New implementations of IPsec and firewall systems may want to interpret the portname option for use in these policy rules, but again should not rely on either port numbers or portnames to indicate a specific service.

The TCP portname option occupies space in the TCP SYN segment. Such space is severely limited in cases where TCP-level security is present, as noted in detail in Section 5.

>> The portname option MUST be protected in the same way that the existing SYN destination port is protected.

For IPsec, this is not an issue because the entire TCP header and payload are protected by all IPsec modes. None of the TCP header is protected by application-layer security, e.g., TLS, so again this is not an issue [RFC2246].

The resulting primary concern is TCP-level security, e.g., TCP/MD5 and its proposed successors [RFC2385] [Bo05] [We06]. TCP/MD5 and [We06] exclude TCP options in their hash calculation; this is confusing, as it fails to protect current critical TCP options such as alternate checksums, window scale, and timestamp options [RFC793] [RFC1323]. [Bo05] allows options to be included or excluded, depending on a header parameter. This document recommends, as per above, that the portname option, as all options, be included in TCP-level protection.

A number of protocols exchange port numbers in-band, notably to coordinate separate concurrent connections, e.g., FTP (file transfer) and SIP (teleconferencing). Because these protocols coordinate the specific port numbers in advance, there is no need for the portname option to indicate the desired service. As a result, it is unlikely that it would be useful to augment these protocols to support the portnames option.
Network address and port translators, known collectively as NATs, not only read TCP ports, but may also translate them [RFC2993]. This interferes with the use of ports for service identification [RFC3234]. The portname option may allow services to be identified behind NATs if NATs are not further extended to translate portname options. It is thus unknown whether the portname option will help restore service identification in the presence of NATs.

TCP connections using the portname option continue to use IP addresses and ports, although both port numbers are typically set arbitrarily. Translation of these ports should not interfere with the operation of NATs, though this has not been verified and is not a design requirement.

4.3. Potential Use in Other Transport Protocols

As noted earlier, the portname option may be a useful addition to a variety of other transport protocols, such as UDP, SCTP and DCCP [RFC768] [RFC2960] [RFC4340]. Adding portname support to SCTP and DCCP should be straightforward because both already have an option space. These are not addressed further in this document, because this focuses on TCP only.

UDP lacks options, so adding support for portnames is not straightforward. One possible approach uses the TCPMUX approach, in which the destination port is fixed to a reserved ‘portname’ port and the NAMESTRING and LENGTH appear in the data (Figure 2). The packet data is located immediately after the NAMESTRING field.

```
  0  7  8 15 16 23 24 31
+--------+--------+--------+--------+
|     Source      |     portname    |
|      Port       |      Port       |
+--------+--------+--------+--------+
|   UDP message   |                 |
|     Length      |    Checksum     |
+--------+--------+--------+--------+
|      | LENGTH | NAMESTRING...
+---------------- ...
```

Figure 2 Potential UDP portname option format

This possible UDP format is viable because UDP has no connection semantics to violate by placing options in the UDP data area. Using a fixed portname port limits the number of outstanding UDP messages to 65,536, but this is less of an issue because UDP has no address/port-
tuple based semantics; any request/response exchange that uses UDP must include its own layer of message identifiers anyway.

4.4. Discussion of Alternative Approaches

The current proposal assumes that the source TCP selects both source and destination port numbers randomly, that the portname option occurs only in SYN and SYN-ACKs, and that the binding of connection to service happens inside the destination at mapping of TCB-to-process. A number of alternative approaches were considered during the development of the approach presented herein. These include:

- A portmapper-like service that returns a specific port number
- Continued demuxing based on the portname option
- Dynamic overwriting of the destination port

The first approach, of returning a specific port number for a service, requires a separate round trip and messages to initiate a connection. We avoid both the additional time and messages in the proposed solution which integrates the lookup in the SYN.

Continued demultiplexing based on portname would violate TCP connection semantics, which indicate that a connection be uniquely identified by the 4-tuple: <src addr><src port><dst addr><dst port>. Although portname demuxing would increase the connection name space, this seems unnecessary as it is already over 4,000,000,000 even between a single pair of host addresses. Finally, this variant incurs the portname NAMESTRING overhead on every message, which seems unnecessarily inefficient. The proposed solution is more efficient and sufficiently increases the utility of the entire current connection name space.

Dynamic overwriting of the destination port would allow late-binding of the destination port. This complicates the connection establishment on the source side, because the SYN-ACK would have a different port pair than the SYN. The primary utility for overwriting the port number would be to facilitate demultiplexing at the receiver, but this is should already include the entire 4-tuple anyway. Overall, this variant seems unnecessarily complex for no real benefit.

5. The Portname Name Space

The portname option requires a new portname name space. IANA already manages an equivalent space, known as the "keyword" name of a port,
which is allocated first-come, first-served, and whose strings typically already include both the protocol name and version number. The portname option is designed to use those same keywords, but renders the need for port number allocation unnecessary.

Portnames need to fit inside the available TCP option space, which provides 40 bytes for options. It is useful to consider that TCP SYN packets may include other options consuming up to 33 bytes, notably:

- 16 bytes of authentication [Bo05] [We06] [To06]
- 4 bytes of MSS
- 10 bytes of timestamp
- 3 bytes of windowscale

This leaves only 7 bytes for the portname option, or 5 bytes for the NAMESTRING. This would accommodate only 20% of existing port names, as port keywords are currently distributed nearly linearly from 2 to 15 bytes long as follows:

```
100% +---------------------------*-
 90% +                        * *
 80% +                       * * *
 70% +                      * * * *
 60% +                     * * * * *
 50% +                   * * * * * *
 40% +                 * * * * * * *
 30% +               * * * * * * * *
 20% +           * * * * * * * * *
 10% +         * * * * * * * * * *
  0%        * * * * * * * * * * *
                  0 0 0 0 0 0 0 0 1 1 1 1 1
                  1 2 3 4 5 6 7 8 9 0 1 2 3 4
```

Port keyword length

Figure 3 Distribution of current port keywords

Note that the most common ports used in the Internet are not necessarily those with the shortest names; although HTTP represents 65% of traffic, services like Gnutella and Napster are also nontrivially represented. We note, however, that the use of TCP authentication is currently deprecated except for routing applications, most of which have short names (e.g., bgp, ldp, msdp).
When non-TCP authentication is used - e.g., IPsec - the available portname name space is over 20 bytes; this accommodates 100% of current names, none of which is longer than 15 bytes.

Portnames MUST be handled as an opaque sequence of bytes; note that this includes zero-valued bytes which some systems interpret as "end of string" characters. To support human-readable names, portnames are represented as UTF-8 [RFC3629].

6. Security Considerations

There are four areas of security which the portname option raises:

1. Interaction with IPsec and firewalls
2. Interaction with TCP/MD5 and other TCP-level security
3. Increased DOS impact

The impact on IPsec and firewalls is discussed in detail in Section 4.2. As noted there, the portname option defeats the assumption that port numbers correspond to specific services, an assumption that was already defeated between consenting hosts. The portname option raises no new vulnerability.

The impact of the portname option on TCP/MD5 and other TCP-layer security services is also discussed in Sections 4.2 and 5. Use of these services without inclusion of TCP options makes all options vulnerable, including the portname option. Because the portname option places the service identifier in this insecure option space, it increases TCP's vulnerability. The appropriate solution would be to use a TCP-level security that covers options, such as [Bo05]. Use of these security options also reduces the space available for the portname option, but this affects only for a limited set of routing protocols that already have short port keywords, and thus should not be an issue.

The use of strings as port identifiers means that TCP SYN segments are longer, requiring more space to buffer while processing. The port indexing is a more complex operation, requiring string lookup rather than 16-bit indexing. These two aspects cause TCP SYN flooding attacks to consume more space and processing resources when the portname option is used. Typical SYN flooding mitigations can be used here [Ed06]. The additional resources incurred by the portname option are minimal, and can be mitigated by separate limits on the rate of portname options processed.
7. IANA Considerations

This document specifies a new TCP option. The KIND for this option must be assigned by IANA prior to this document’s issuance as an RFC.

This document requires IANA to manage a new name space of "TCP Portnames". This name space should be initially seeded with the "keywords" of current TCP ports. This namespace MUST include any UTF-8 encoded string. Although the current port keywords are limited to 15 characters, portnames are limited only by available TCP option space.

IANA should allocate a TCP destination port for the portname service, to allow applications to bind to that port and then receive on any incoming destination port. This is the port space equivalent of INADDR_ANY, and is thus called INPORT_ANY; the value "0" is suggested.

[AUTHOR’S NOTE: if the UDP variant is included, the destination port should be INPORT_ANY; TCP connections to INPORT_ANY that lack the TCP portname option MUST be rejected.]

8. Conclusions

<to be completed>

9. Acknowledgments

This work was inspired by discussions on the IETF mailing list, notably by suggestions by Keith Moore and Noel Chiappa. Bob Braden noted some of the origins of the named service concept.
10. References

10.1. Normative References


10.2. Informative References


[IANA] Internet Assigned Numbers Authority, www.iana.org


Author’s Addresses

Joe Touch
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292-6695
U.S.A.

Phone: +1 (310) 448-9151
Email: touch@isi.edu
URL: http://www.isi.edu/touch

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.
Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

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