PCEP TCP Port Restriction (RFC 5440)

R. Casellas, R. Martínez, R. Muñoz
Optical Networking Area
Centre Tecnològic de Telecomunicacions de Catalunya (CTTC)

80 IETF Meeting / PCE Working Group / Prague, March 2011

Introduction and Context

- PCEP RFC restricts TCP source and destination ports to registered port 4189
  - Rationale not clear (*at least to us…)*
    - See pce@ietf Mon, 2 Jun 2008 16:03:02 -0400 (http://www.ietf.org/mail-archive/web/pce/current/msg01634.html)
    - *"It might be claimed that two PCC processes might exist on a single host/router, but no usage scenario has been found"* → A PCE "host" connecting to peers will "behave" as different PCCs.
    - It does not improve security.
    - It does not ease configuration / administration.
    - It is not simpler to implement
  - A "PCE" may end up having several TCP/PCEP adjacencies, either as server / client

- Seems contrary to "common practice"
  - TCP servers "listen" to well known / registered ports (not necessarily < 1024)
    - BGP (179) / HTTP (80) / IRC (6667) / TELNET (23) / FTP ….
  - TCP clients rely on a O.S. selected random port

- BUT, seems "perfectly" valid
  - assuming coupled restriction of only one TCP connection between peers.
    - TCP "tuple" (src address, src port 4189, dst address, dst port 4189 [protocol]) is unique

Practical Deployments

- It is common to use e.g. a “loopback address” / Node ID / Router ID / PCE ID / … to identify a “multi-homed” host

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet 10.0.50.1/32 brd 10.0.50.1 scope global lo:0
```

- Most common operating systems use BSD sockets API – or similar
  - In order to use e.g. loopback address, processes “bind” sockets to it
    - Forces “local source address”
  - Alternatively, one can bind to INADDR_ANY
    - means “connections to any address are ok” (server) or “use outgoing interface address” (client).
  - A scenario where a system process binds the same IP address / port in both TCP server / client mode is not common

Centre Tecnològic de Telecomunicacions de Catalunya

Practical Issue

- It is not possible (in a portable way), to "bind" or associate several sockets to the same (local address, local port) pair
  - **Use case**: after the first "listening" socket, the PCE would, for each persistent peer (i) create a socket SOCK_STREAM, (ii) bind it to nodeid/4189 and (iii) connect to remote peer/4189
    - Linux ip(7) "Only one IP socket may be bound to any given local (address, port) pair".
  - PCEP requires duplicate bindings at the socket level.
    - Some systems may implement SO_REUSEPORT / SO_REUSEADDR (1) socket option which could allow that, although it main use is to bind multicast addresses, or enable some sort of load sharing amongst TCP listeners in several threads (this feature may only be supported only for UDP sockets) → interpretation varies.
  - *In practice this seems to affect GNU/Linux in particular and windows / *BSD seem to work ok, but e.g. java programs behave differently on different platforms.*

Centre Tecnològic de Telecomunicacions de Catalunya

Recommended Errata (draft)

5. Transport Protocol

PCEP operates over TCP using a registered TCP port (4189). This allows the requirements of reliable messaging and flow control to be met without further protocol work. ~~All PCEP messages MUST be sent using the registered TCP port for the source and destination TCP port~~.

9.1. TCP Port

PCEP has been registered as TCP port 4189.

10.7.1

- o PCEP uses a single registered port for all communications. The PCE SHOULD listen for TCP connections only ~~on ports where communication is expected~~.

- o The PCE MAY implement an access list to immediately reject (or discard) TCP connection attempts from unauthorized PCCs.

- o The PCE SHOULD NOT allow parallel TCP connections from the same PCC on the PCEP-registered port.

5. Transport Protocol

PCEP operates over TCP using a registered TCP port (4189). This allows the requirements of reliable messaging and flow control to be met without further protocol work. A PCE MUST listen for incoming connections at the registered port and a *PCC SHOULD use the registered port as source port, but MAY use any source port.*

9.1. TCP Port

PCEP has been registered as TCP port 4189.

10.7.1

- o PCEP uses a single registered port for all communications. The PCE SHOULD listen for TCP connections only on the PCEP-registered port.

- o The PCE MAY implement an access list to immediately reject (or discard) TCP connection attempts from unauthorized PCCs.

- o The PCE SHOULD NOT allow parallel TCP connections from the same PCC on the PCEP-registered port.

Relax TCP Port Restriction while allowing currently conforming PCEP implementations to operate without exchange (except those that enforce remote client source port)

Centre Tecnològic de Telecomunicacions de Catalunya

Backup: sketch of issue

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/ioctl.h>
#include <net/if.h>

int one = 1;
int portno = 4189;

void error(const char *msg)
{
	perror(msg);
	exit(EXIT_FAILURE);
}

void get_iface_ipv4(char *ifname, struct sockaddr *addr)
{
	struct ifreq ifr;
	int skfd = socket(AF_INET, SOCK_DGRAM, 0);
	strcpy(ifr.ifr_name, ifname);
	ioctl(skfd, SIOCGIFADDR, &ifr);
	*addr = ifr.ifr_addr;
	close(skfd);
}
```

```
int client (struct sockaddr_in* local_addr)
{
	int sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &one, sizeof(one)) < 0)
		error("SO_REUSEADDR client");
	if (bind(sockfd, (struct sockaddr *)local_addr, sizeof(struct sockaddr_in)) < 0)
		error("bind client");
	return sockfd;
}

int main(int argc, char *argv[])
{
	struct sockaddr_in local_addr;
	memset(&local_addr, 0, sizeof(struct sockaddr_in));
	get_iface_ipv4("eth0", (struct sockaddr*) &local_addr);
	local_addr.sin_family = AF_INET;
	local_addr.sin_port = htons(portno);

	int sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &one, sizeof(one)) < 0)
		error("SO_REUSEADDR server");
	if (bind(sockfd, (struct sockaddr *) &local_addr, sizeof(local_addr)) < 0)
		error("ERROR on binding");
	if (listen(sockfd, 10) <0)
		error("ERROR on listen");

	int c1 = client (&local_addr);
	return 0;
}
```