people.csail.mit.edu

Service Discovery Protocol

So far this chapter has shown how to detect nearby Bluetooth device and establish the two main types of data transport connections, all using fixed Bluetooth address and port numbers that were determined at design time. As mentioned in Section 3.4, this is not a recommended practice in general.

Dynamically allocating port numbers and using the Service Discovery Protocol (SDP) to search for and advertise services is a simple process in PyBluez. The `get_available_port` method finds available L2CAP and RFCOMM ports, `advertise_service` advertises a service with the local SDP server, and `find_service` searches Bluetooth devices for a specific service.

```
bluetooth.get_available_port( protocol )
```

`get_available_port` returns the first available port number for the specified protocol. Currently, only the RFCOMM and L2CAP protocols are supported. `get_available_port` only returns a port number, and does not actually reserve any resources, so it is possible that the availability changes between the time we call `get_available_port` and `bind`. If this happens, `bind` will simply raise a `BluetoothException`.

```
bluetooth.advertise_service( sock, name, uuid )
bluetooth.stop_advertising( sock )
bluetooth.find_service( name = None, uuid = None, bdaddr = None )
```

These three methods provide a way to advertise services on the local Bluetooth device and search for them on one or many remote devices. `advertise_service` takes a socket that is bound and listening, a service name, and a UUID as input parameters. PyBluez requires the socket to be bound and listening because there is no point in advertising a service that does not exist yet. The UUID must always be a string of the form ``xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" or ``xxxxxxxx" or ``xxxx", where each 'x' is a hexadecimal digit. The service will be advertised as long as the socket is open, or until a call is made to `stop_advertising`, specifying the advertised socket.

`find_service` can search either a single device or all nearby devices for a specific service. It looks for a service with name and UUID that match `name` and `uuid`, at least one of which must be specified.. If `bdaddr` is `None`, then all nearby devices will be searched. In the special case that ``localhost" is used for `bdaddr`, then the locally advertised SDP services will be searched. Otherwise, the function search the

services provided by the Bluetooth device with address `bdaddr`.

On return, `find_service` returns a list of dictionaries. Each dictionary contains information about a matching service and has the entries ``host", ``name", ``protocol", and ``port". *host* indicates the address of the device advertising the service, *name* is the name of the service advertised, *protocol* will be either ``L2CAP", ``RFCOMM", or ``UNKNOWN", and *port* will be the port number that the service is operating on. Typically, only the protocol and port number are needed to connect. Example 3-6 and Example 3-7 show the RFCOMM client and server from the previous section modified to use dynamic port assignment and SDP to advertise and discover services.

Example 3-6. rfcomm-server-sdp.py

```
import bluetooth

server_sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )

port = bluetooth.get_available_port( bluetooth.RFCOMM )
server_sock.bind(("",port))
server_sock.listen(1)
print "listening on port %d" % port

uuid = "1e0ca4ea-299d-4335-93eb-27fcfe7fa848"
bluetooth.advertise_service( server_sock, "FooBar Service", uuid )

client_sock,address = server_sock.accept()
print "Accepted connection from ",address

data = client_sock.recv(1024)
print "received [%s]" % data

client_sock.close()
server_sock.close()
```

Here, the server from Example 3-2 is modified to use `get_available_port` and `advertise_service`. The UUID ``1e0ca4ea-299d-4335-93eb-27fcfe7fa848" is used to identify the ``FooBar service". The client from Example 3-3 is modified to use `find_service` to search for the the server, and connects to the first server found. The client makes an implicit assumption that the transport protocol used by the server is RFCOMM.

Example 3-7. rfcomm-client-sdp.py

```
import sys
import bluetooth

uuid = "1e0ca4ea-299d-4335-93eb-27fcfe7fa848"
service_matches = bluetooth.find_service( uuid = uuid )

if len(service_matches) == 0:
    print "couldn't find the FooBar service"
    sys.exit(0)

first_match = service_matches[0]
port = first_match["port"]
name = first_match["name"]
host = first_match["host"]

print "connecting to \"%s\" on %s" % (name, host)

sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )
sock.connect((host, port))
sock.send("hello!!")
sock.close()
```