Service Provider Interface for 1394PWG Transport Protocol

Brian Batchelder

Hewlett-Packard

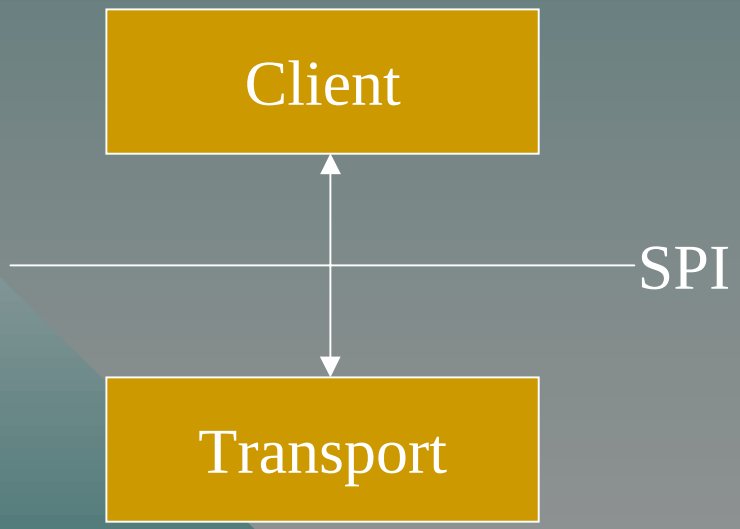
October 2, 1998

Definitions

Service Provider Interface

- The Service Provider Interface (SPI) describes the services provided by the transport protocol to its clients.
- It does *not* describe a required Application Programming Interface (API) for the transport protocol.

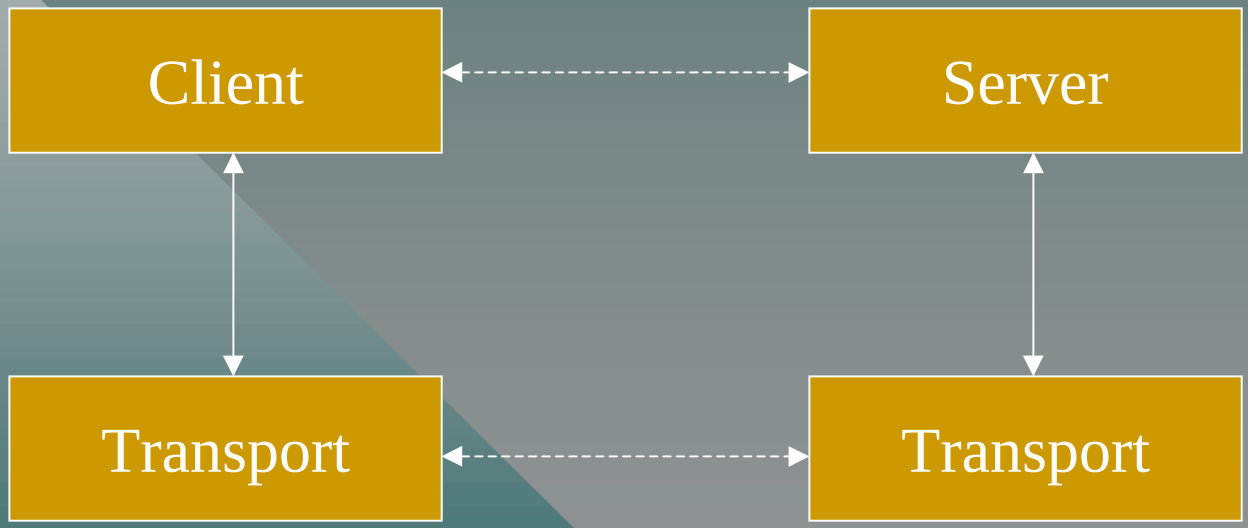
Transport Clients



Clients and Servers

- The clients of the transport protocol are applications that inter-communicate.
- One of these client applications provides services and is known as the “server”.
- The client application that uses the services is known as the “client”.

Clients and Servers



PWG Clients and Servers

- In the PWG world, typically:
  - Clients reside in a host (printer drivers, scanner applications, fax utilities, status monitors)
  - Servers reside in a device (print parsers, scan engines, fax engines, status & control engines)
- Typical world is not entire world, nor is it necessarily the future world.

Requirements

Client Requirements

- Clients want to:
  - Discover and connect to servers
  - Transfer data across the connection
  - Disconnect from servers
  - Monitor errors and asynchronous events

Server Requirements

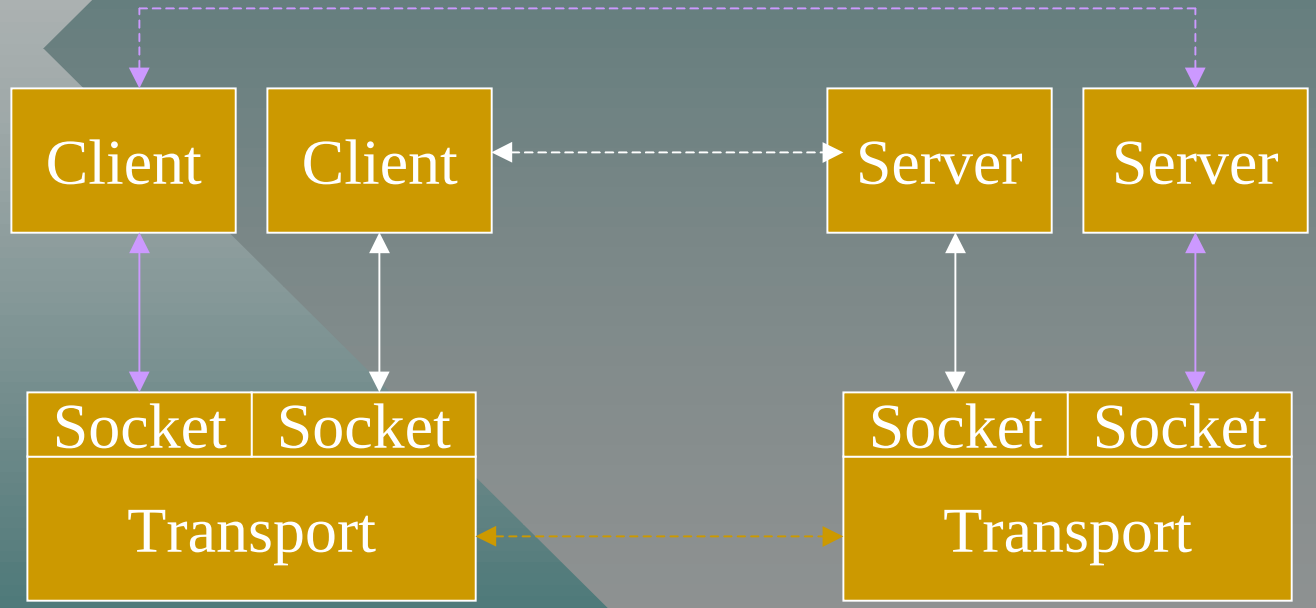
- Servers want to:
  - Identify themselves with a service name
  - Accept connections from clients
  - Transfer data across the connection
  - Disconnect from clients?
  - Monitor errors and asynchronous events

Service Provider Interface

Socket-based Interface

- A socket-based interface is a standard API for transport protocols.
- A socket is simply a data structure referring to an application.
  - A single application can have more than one socket.
  - A socket can refer to only one application.
- BSD and Winsock-2 are well-known socket interfaces

Sockets


Client
Client
Server
Server
Socket
Socket
Socket
Socket
Transport
Transport

Services

- socket()
- bind()
- listen()
- connect()
- accept()
- send()
- recv()
- shutdown()
- closesocket()
- setsockopt()
- getsockopt()

Services (detailed)

- socket()
  - Create an endpoint for communication and return a socket descriptor.
- bind()
  - Assign a local name to an unnamed socket.
- listen()
  - Listen for incoming connections on a specified socket.

Services (detailed) (cont.)

- connect()
  - Initiate a connection on the specified socket.
- accept()
  - An incoming connection is acknowledged and associated with an immediately created socket. The original socket is returned to the listening state.

Services (detailed) (cont.)

- send()
  - Send data to a connected socket.
- recv()
  - Receive data from a connected or unconnected socket.

Services (detailed) (cont.)

- shutdown()
  - Shut down part of a full-duplex connection.
- closesocket()
  - Remove a socket from the per-process object reference table.

Services (detailed) (cont.)

- setsockopt()
  - Store options associated with the specified socket.
- getsockopt()
  - Retrieve options associated with the specified socket.

SPI Usage

Clients

- Clients establish a socket by calling socket().
- Once the socket has been allocated, the client can request a connection by calling connect(). When connect() completes, the connection is open.

Clients (cont.)

- Once a connection is open, clients can use send() and receive() to transfer data across the open connection.
- The client can close the connection using shutdown().
- When the client is finished using a socket, it releases the socket by calling closesocket().

Servers

- Servers first establish a socket by calling socket() and then bind their service name to that socket by calling bind().
- Once the socket has been allocated, servers call listen() to establish an incoming request queue, and then call accept() to wait for a connection request.

Servers (cont.)

- When a client requests a connection, accept() will complete with a new socket for the now open connection. If the server can accept more concurrent connections, it may call listen() again with the original socket.

Servers (cont.)

- Once a connection is open, servers can use send() and receive() to transfer data across the open connection.
- The server can close the connection using shutdown()
- When the server is finished using a socket, it releases the socket by calling closesocket().

Acknowledgements

- Much of the information in this presentation comes from the Winsock-2 API document revision 2.2.0. The current revision is available at Intel (http://www.intel.com/IAL/winsock2/) and Microsoft web-sites.