Elementary Sockets

Lecture 3 (C)

Topic Outline

- Introduction to TCP and UDP sockets
- TCP and UDP client-server programs
- SCTP association
- I/O multiplexing and non-blocking I/O
- **Socket options**
- Remote Method Invocation
- Name and address conversions

Socket Options

- Java provides a type-safe way to set options.

- Each socket class has a get/set method for each option it supports, taking and returning the appropriate type.

- **void setOption(int optID, Object val) throws SocketException;**
- **Object getOption(int optID) throws SocketException;**

Supported Socket Options in Java

- **TCP_NODELAY**
  - Disable Nagle's algorithm.
  - Valid for (client) Sockets.
- **SO_LINGER**
  - Specify a linger-on-close timeout.
  - Valid for (client) Sockets.
- **SO_TIMEOUT**
  - Specify a timeout on blocking socket operations. (Don't block forever!)
  - Valid for all sockets: Socket, ServerSocket, DatagramSocket.

Supported Socket Options in java.net.SocketOptions Interface

- **SO_BINDADDR**
  - Fetch the local address binding of a socket.
  - Valid for Socket, ServerSocket, DatagramSocket.
- **SO_REUSEADDR**
  - Enable reuse address for a socket.
  - Valid for Socket, ServerSocket, DatagramSocket.
- **SO_BROADCAST**
  - Enables a socket to send broadcast messages.
  - Valid for DatagramSocket.

Supported Socket Options in java.net.SocketOptions Interface

- **SO_SNDBUF**
  - Set a hint the size of the underlying buffers for outgoing network I/O.
  - Valid for all sockets: Socket, ServerSocket, DatagramSocket.
- **SO_RCVBUF**
  - Get the size of the buffer actually used by the platform when receiving in data on this socket.
  - Valid for all sockets: Socket, ServerSocket, DatagramSocket.
- **SO_KEEPALIVE**
  - Turn on socket keepalive.
  - Valid for Socket.

Supported Socket Options in java.net.SocketOptions Interface

- **SO_OOBINLINE**
  - Enable inline reception of TCP urgent data.
  - Valid for Socket.
- **IP_MULTICAST_IF**
  - Specify the outgoing interface for multicast packets (on multihomed hosts).
  - Valid for MulticastSockets.
- **IP_MULTICAST_LOOP**
  - Enables or disables local loopback of multicast datagrams.
  - Valid for MulticastSocket.
- **IP_TOS**
  - Sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.
  - Valid for Socket, DatagramSocket

TCP_NODELAY

- Disables **Nagle's Algorithm**
- Valid for (client) Sockets.

- java.net.Socket
  - **`public void setTcpNoDelay(boolean on) throws SocketException;`**

TCP_NODELAY

- Nagle’s Algorithm
  - Named after John Nagle, is a means of improving the efficiency of TCP/IP networks by reducing the number of packets that need to be sent over the network.
  - Nagle's document, **Congestion Control in IP/TCP Internetworks (RFC 896)** describes the "small packet problem“ where an application repeatedly emits data in small chunks, frequently only 1 byte in size.
  - Since TCP packets have a 40 byte header (20 bytes for TCP, 20 bytes for IPv4), this results in a 41 byte packet for 1 byte of useful information, a huge overhead.
  - Often occurs in Telnet sessions, where most keypresses generate a single byte of data that is transmitted immediately.
  - Nagle's algorithm works by combining a number of small outgoing messages, and sending them all at once. Specifically, as long as there is a sent packet for which the sender has received no acknowledgment, the sender should keep buffering its output until it has a full packet's worth of output, so that output can be sent all at once.

SO_LINGER

- Specify a linger-on-close timeout.
- Valid for (client) Sockets.

- java.net.Socket
  - `public void setSoLinger(boolean on, int linger) throws SocketException;`

SO_LINGER

- Used to specify how the close() method affects socket using a connection-oriented protocol (i.e. TCP/IP, not UDP)
- On invoking the close() method, there are several things that can happen, depending on the state of SO_LINGER:
  - **Linger set to false (default)**
    - No more receives or sends can be issued on the socket. The contents of the socket send buffer are sent to the other end. Data in socket receive buffer is discarded. The close() method returns immediately, performing these activities in the background.
  - **Linger set to true and linger time==0**
    - No more receives or sends can be issued on the socket. The socket send and receive buffers are both discarded. This means that if the OS has data internally for the socket this data is not sent and not received by the other end. The close() method returns immediately and clears the send buffer in background.
  - **Linger set to true and linger time!=0**
    - No more receives or sends can be issued on the socket. Data in socket send buffer is sent and data in the receive buffer is discarded. If the linger time expires an exception will occur. The close() method will block for a maximum of linger seconds or until data has been sent and acknowledged at the TCP level.

SO_TIMEOUT

- Specify a timeout on blocking socket operations. (Don't block forever!)
- Valid for all sockets: Socket, ServerSocket, DatagramSocket.

- java.net.Socket
  - **`public void setSoTimeout(int timeout) throws SocketException`**

- Enable/disable SO_TIMEOUT with the specified timeout, in milliseconds. With this option set to a non-zero timeout, a read() call on the InputStream associated with this Socket will block for only this amount of time.
- If the timeout expires, a java.net.SocketTimeoutException is raised, though the Socket is still valid. The option must be enabled prior to entering the blocking operation to have effect. The timeout must be > 0. A timeout of zero is interpreted as an infinite timeout.

SO_BINDADDR

- Fetch the local address binding of a socket.
- Valid for Socket, ServerSocket, DatagramSocket.

- java.net.SocketOptions

- Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed).
- The default local address of a socket is INADDR_ANY, meaning any local address on a multi-homed host.
- A multi-homed host can use this option to accept connections to only one of its addresses (in the case of a ServerSocket or DatagramSocket), or to specify its return address to the peer (for a Socket or DatagramSocket). The parameter of this option is an InetAddress.

SO_REUSEADDR

- Enable reuse address for a socket.
- Valid for Socket, ServerSocket, DatagramSocket.

- java.net.Socket
  - **`public void setReuseAddress(boolean on) throws SocketException`**

- When a TCP connection is closed the connection may remain in a timeout state for a period of time after the connection is closed (typically known as the TIME_WAIT state or 2MSL wait state).
- For applications using a well known socket address or port it may not be possible to bind a socket to the required SocketAddress if there is a connection in the timeout state involving the socket address or port.
- Enabling SO_REUSEADDR prior to binding the socket using bind(SocketAddress) allows the socket to be bound even though a previous connection is in a timeout state.

SO_BROADCAST

- Enables a socket to send broadcast messages.
- Valid for DatagramSocket.

- java.net.DatagraSocket
  - **`public void setBroadcast(boolean on) throws SocketException`**

SO_SNDBUF

- Set a hint the size of the underlying buffers for outgoing network I/O.
- Valid for all sockets: Socket, ServerSocket, DatagramSocket.

- java.net.Socket
  - **`public void setSendBufferSize(int size) throws SocketException`**

- Sets the SO_SNDBUF option to the specified value for this Socket.
- The SO_SNDBUF option is used by the platform's networking code as a hint for the size to set the underlying network I/O buffers.

SO_RCVBUF

- Get the size of the buffer actually used by the platform when receiving in data on this socket.
- Valid for all sockets: Socket, ServerSocket, DatagramSocket.

- java.net.Socket
  - **`public void setReceiveBufferSize(int size) throws SocketException`**

- Sets the SO_RCVBUF option to the specified value for this Socket. The SO_RCVBUF option is used by the platform's networking code as a hint for the size to set the underlying network I/O buffers.
- Increasing the receive buffer size can increase the performance of network I/O for high-volume connection, while decreasing it can help reduce the backlog of incoming data.

SO_KEEPALIVE

- Turn on socket keepalive.
- Valid for Socket.

- java.net.Socket
  - `public void setKeepAlive(boolean on) throws SocketException`

SO_OOBINLINE

- Enable inline reception of TCP urgent data.
- Valid for Socket.

- java.net.Socket
  - **`public void setOOBInline(boolean on) throws SocketException`**

- Enable/disable OOBINLINE (receipt of TCP urgent data) By default, this option is disabled and TCP urgent data received on a socket is silently discarded.
- If the user wishes to receive urgent data, then this option must be enabled. When enabled, urgent data is received inline with normal data.

IP_MULTICAST_IF

- Specify the outgoing interface for multicast packets (on multihomed hosts).
- Valid for MulticastSockets.

- java.net.MulticastSocket
  - `public void setInterface(InetAddress inf) throws SocketException`

- Set the multicast network interface used by methods whose behavior would be affected by the value of the network interface. Useful for multihomed hosts.

IP_MULTICAST_LOOP

- Enables or disables local loopback of multicast datagrams.
- Valid for MulticastSocket.

- java.net.MulticastSocket
  - **`public void setLoopbackMode(boolean disable) throws SocketException`**

- Disable/Enable local loopback of multicast datagrams The option is used by the platform's networking code as a hint for setting whether multicast data will be looped back to the local socket.

IP_TOS

- Sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.
- Valid for Socket, DatagramSocket

- java.net.Socket
  - **`public void setTrafficClass(int tc) throws SocketException`**

- Sets traffic class or type-of-service octet in the IP header for packets sent from this Socket. As the underlying network implementation may ignore this value applications should consider it a hint.
- The tc must be in the range 0 <= tc <= 255 or an IllegalArgumentException will be thrown.

References

- http://docs.oracle.com/javase/7/docs/api/java/net/StandardSocketOptions.html
- http://docs.oracle.com/javase/7/docs/api/java/net/Socket.html