A UDP Interface

Allan Heydon

DRAFT — December 11, 1998 — DRAFT

1 UDP.i3

The User Datagram Protocol (UDP) is a connectionless protocol for sending data packets (or *datagrams*) over the internet. UDP runs on top of the Internet Protocol (IP). It provides a simple datagram service that guarantees the integrity of delivered packets. However, unlike the Transmission Control Protocol (TCP), which also runs on IP, UDP does not guarantee reliable delivery of packets: packets may be dropped or duplicated, and any corrupted packets are simply discarded. Nor does UDP provide any sequencing guarantees.

Like TCP, each UDP endpoint is identified by the IP address of a host together with a port number (see the IP interface for a description of IP endpoints). Hence, datagrams may be addressed to different processes (listening on different ports) that are running on the same host.

The complete UDP protocol is documented in Internet RFC 768 [1].

```
INTERFACE UDP;

IMPORT IP, Thread;

TYPE
  Datagram = RECORD
    other: IP.Endpoint;
    len: CARDINAL;
    bytes: REF ARRAY OF CHAR;
  END;
```

A `UDP.Datagram` represents a packet of data. If `d` is a `Datagram`, then `d.other` is the IP endpoint to which the packet will be sent, or from which it was received, `d.len` is the length of the packet, and `d.bytes` contains the packet contents. A valid packet has `d.bytes # NIL` and `d.len <= NUMBER(d.bytes^)`, in which case the bytes of the packet are in `d.bytes[0..(d.len-1)]`.

```
EXCEPTION Timeout;
```

The `Timeout` exception may be raised to indicate a timeout on a blocking UDP operation. The UDP implementation may also raise the `IP.Error` exception; arguments to this exception will be one of the predefined values `IP.NoResources` or `IP.PortBusy`, or a message indicating that an unexpected error occured.

```
TYPE
```

```
    T <: Public;
    Public = OBJECT METHODS
      init(myPort: IP.Port; myAddr := IP.NullAddress): T
        RAISES {IP.Error};
      send(READONLY d: Datagram): INTEGER
        RAISES {IP.Error};
      sendText(READONLY other: IP.Endpoint; t: TEXT): INTEGER
        RAISES {IP.Error};
      receive(VAR (*INOUT*) d: Datagram; timeout: LONGREAL := -1.0d0)
        RAISES {Timeout, IP.Error, Thread.Alerted};
      close()
        RAISES {IP.Error};
    END;

  END UDP.
```

A `UDP.T` is a handle on a socket for receiving datagrams on a particular port.

The expression `NEW(UDP.T).init(myPort)` evaluates to a new UDP handle for listening on port `myPort`. If the program is being run on a machine with multiple IP addresses, an `IP.Address` argument can also be passed in the optional `myAddr` parameter to indicate which network interface will be used by this handle to send and receive datagrams.

If a UDP handle is closed using the `close` method, the `init` method can be called again to initialize the handle on a different port/address. All methods of a `UDP.T` other than `init` require that the UDP handle has been initialized; `init` requires that the handle is new or closed.

The descriptions of the remaining methods assume that `udp` denotes an initialized `UDP.T`.

The call `udp.send(d)` sends the datagram `d`, returning the number of bytes that were successfully sent.

The call `udp.sendText(other, t)` sends the contents of the text `t` to the endpoint `other`, returning the number of characters of `t` that were successfully sent.

The call `udp.receive(d)` blocks until a datagram is sent to `udp`'s port/address. On entry, `d.bytes` should point to an array of characters large enough to contain incoming datagrams. On exit, `d.other` is set to the endpoint from which the datagram was sent, `d.len` is set to the length of the received datagram, and the contents of the datagram are written into `d.bytes[0..(d.len)-1]`. If the packet that was received is larger than the initial value of `d.len`, the packet is truncated to `d.len` bytes.

If a non-negative value is supplied for `timeout`, the method will raise `Timeout` if `timeout` seconds elapse without a packet being received. Negative values of `timeout` indicate an indefinite wait. If the calling thread is alerted before a packet is received, `Thread.Alerted` is thrown.

The call `udp.close` closes `udp`. Invoking any methods other than `init` on a closed handle results in a checked run-time error.

References

[1] J. Postel. Internet RFC 768: User Datagram Protocol, August 1980. Available on the world-wide web at "http://www.freesoft.org/CIE/RFC/768/".