{-# LANGUAGE TypeSynonymInstances #-}

module Network.UDP

( DataPacket(..)

, openBoundUDPPort

, openListeningUDPPort

, pingUDPPort

, sendUDPPacketTo

, recvUDPPacket

, recvUDPPacketFrom

) where

import qualified Data.ByteString as Strict (ByteString, concat, singleton)

import qualified Data.ByteString.Lazy as Lazy (ByteString, toChunks, fromChunks)

import Data.ByteString.Char8 (pack, unpack)

import Network.Socket hiding (sendTo, recv, recvFrom)

import Network.Socket.ByteString (sendTo, recv, recvFrom)

-- Type class for converting StringLike types to and from strict ByteStrings

class DataPacket a where

toStrictBS :: a -> Strict.ByteString

fromStrictBS :: Strict.ByteString -> a

instance DataPacket Strict.ByteString where

toStrictBS = id

{-# INLINE toStrictBS #-}

fromStrictBS = id

{-# INLINE fromStrictBS #-}

openBoundUDPPort :: String -> Int -> IO Socket

openBoundUDPPort uri port = do

s <- getUDPSocket

bindAddr <- inet\_addr uri

let a = SockAddrInet (toEnum port) bindAddr

bindSocket s a

return s

pingUDPPort :: Socket -> SockAddr -> IO ()

pingUDPPort s a = sendTo s (Strict.singleton 0) a >> return ()