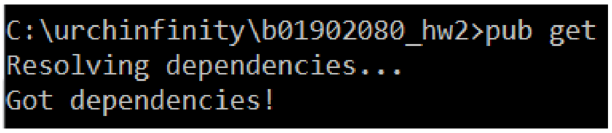
Reliable Data Transfer on UDP

B01902080 資工三 王于青

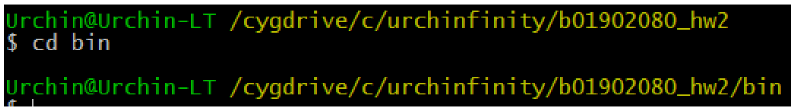
The Program

- Execution environment: Windows (Vista, 7, 8), Linux, Mac OS X
- Required sources: Dart SDK (https://www.dartlang.org/)
- Execution instructions:
  1. Pub get to get required packages.

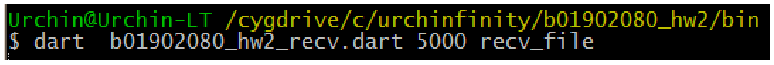


```
C:\urchinfinity\b01902080_hw2>pub get
Resolving dependencies...
Got dependencies!
```

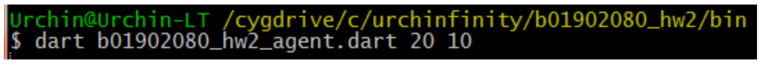
  2. Run b01902080_hw2_recv.dart, b01902080_hw2_agent.dart, b01902080_hw2_send.dart in bin directory.



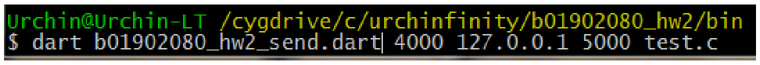
```
Urchin@Urchin-LT /cygdrive/c/urchinfinity/b01902080_hw2
$ cd bin

Urchin@Urchin-LT /cygdrive/c/urchinfinity/b01902080_hw2/bin
```



```
Urchin@Urchin-LT /cygdrive/c/urchinfinity/b01902080_hw2/bin
$ dart  b01902080_hw2_recv.dart 5000 recv_file
```



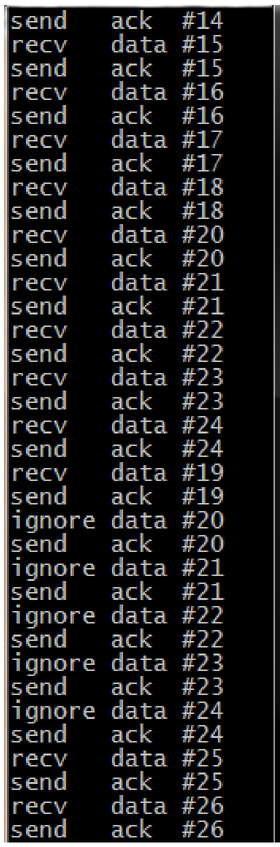
```
Urchin@Urchin-LT /cygdrive/c/urchinfinity/b01902080_hw2/bin
$ dart b01902080_hw2_agent.dart 20 10
```



```
Urchin@Urchin-LT /cygdrive/c/urchinfinity/b01902080_hw2/bin
$ dart b01902080_hw2_send.dart 4000 127.0.0.1 5000 test.c
```

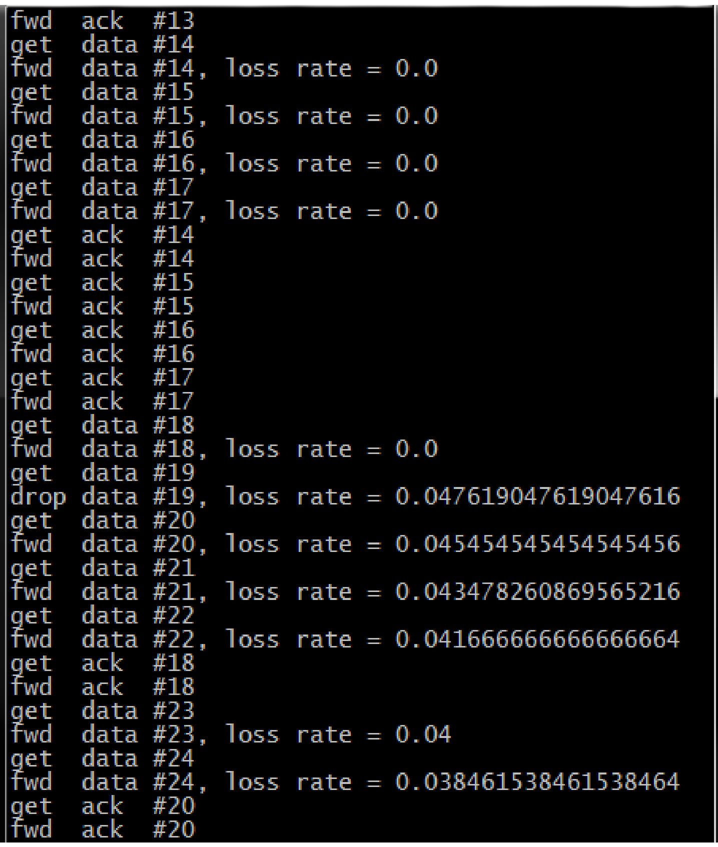
- Result

b01902080_hw2_recv.dart



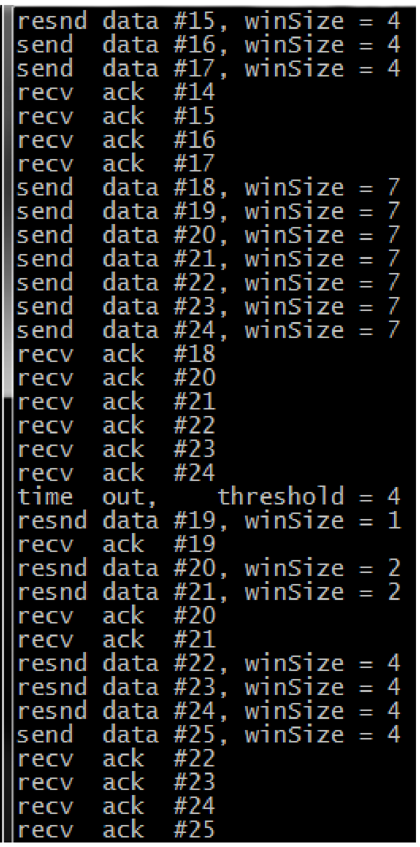
```
send    ack  #14
recv    data #15
send    ack  #15
recv    data #16
send    ack  #16
recv    data #17
send    ack  #17
recv    data #18
send    ack  #18
recv    data #20
send    ack  #20
recv    data #21
send    ack  #21
recv    data #22
send    ack  #22
recv    data #23
send    ack  #23
recv    data #24
send    ack  #24
recv    data #19
send    ack  #19
ignore  data #20
send    ack  #20
ignore  data #21
send    ack  #21
ignore  data #22
send    ack  #22
ignore  data #23
send    ack  #23
ignore  data #24
send    ack  #24
recv    data #25
send    ack  #25
recv    data #26
send    ack  #26
```

b01902080_hw2_agent.dart



```
fwd  ack  #13
get  data #14
fwd  data #14, loss rate = 0.0
get  data #15
fwd  data #15, loss rate = 0.0
get  data #16
fwd  data #16, loss rate = 0.0
get  data #17
fwd  data #17, loss rate = 0.0
get  ack  #14
fwd  ack  #14
get  ack  #15
fwd  ack  #15
get  ack  #16
fwd  ack  #16
get  ack  #17
fwd  ack  #17
get  data #18
fwd  data #18, loss rate = 0.0
get  data #19
drop data #19, loss rate = 0.047619047619047616
get  data #20
fwd  data #20, loss rate = 0.045454545454545456
get  data #21
fwd  data #21, loss rate = 0.043478260869565216
get  data #22
fwd  data #22, loss rate = 0.041666666666666664
get  ack  #18
fwd  ack  #18
get  data #23
fwd  data #23, loss rate = 0.04
get  data #24
fwd  data #24, loss rate = 0.038461538461538464
get  ack  #20
fwd  ack  #20
```

b01902080_hw2_send.dart



```
resnd data #15, winSize = 4
send  data #16, winSize = 4
send  data #17, winSize = 4
recv  ack  #14
recv  ack  #15
recv  ack  #16
recv  ack  #17
send  data #18, winSize = 7
send  data #19, winSize = 7
send  data #20, winSize = 7
send  data #21, winSize = 7
send  data #22, winSize = 7
send  data #23, winSize = 7
send  data #24, winSize = 7
recv  ack  #18
recv  ack  #20
recv  ack  #21
recv  ack  #22
recv  ack  #23
recv  ack  #24
time  out,      threshold = 4
resnd data #19, winSize = 1
recv  ack  #19
resnd data #20, winSize = 2
resnd data #21, winSize = 2
recv  ack  #20
recv  ack  #21
resnd data #22, winSize = 4
resnd data #23, winSize = 4
resnd data #24, winSize = 4
send  data #25, winSize = 4
recv  ack  #22
recv  ack  #23
recv  ack  #24
recv  ack  #25
```

Functions Achieved

- Reliable Data Transfer on UDP Socket
  1. Sequence number:
     Divide file content into fixed size packets.
     Take the order of each packet as its sequence number.
  2. Acknowledge:
     Receiver takes data's sequence number as acknowledge and send it back to sender.
  3. Time out:
     Set a timer for each sent packet.
     When a timer timeouts, add the sequence # of the timeout packet to resent list.
     Also add the sequence # of the rest packets to resent list.
  4. Retransmission:
     Use a packet head flag to trace the first packet to send in each congestion window.
     If the packet is in resent list, print ″resend", otherwise print "send".
- Congestion Control
  Initialize the congestion window size and threshold to default values.
  When receiving an ack, check if sender is in slow start stage and accumulate the window size correspondingly.
  If any packet timeouts, set threshold to $\boxed{\text{cwnd/2}}$ and window size to $\boxed{1}$.
- Buffer Overflow
  When receiving a packet, use $\boxed{\text{alignSequence\# = sequence\# \% bufferLength}}$ to store the packet content in buffer in correct order.
  Set ackRange (first required packet ~ first required packet + bufferLength) as expected range:
  1. If recv_ack < ackRange, send ack back only.
  2. If recv_ack >= ackRange, drop the packet without send ack back.
  3. If recv_ack is in ackRange, read the content if receiver did not receive the packet before, ignore otherwise. Send ack back to sender.

  When receiver drops a packet, it checks if the buffer is full and flush buffer to file.
- Loss Rate Control
  User can specify stable stage (number of stages not to drop packets after start) and loss rate. Agent starts to forward or drop received packet randomly after stable stage.
  The final loss rate may be close to the ideal loss rate if transmitted file is big enough.

Challenging Issues & Solutions

Q: Dart 的 socket API 會去掉 packet 結尾的空白、換行字元，所以最後收到的檔案不完整。
A: 我除了用逗號分開 packet 中的資訊，也在 data 結尾加上逗號代表字串結束。

Q: Sender/Receiver 間 packet 的最後目的地(Receiver/Sender)與真正連線對象(Agent)不同。
A: 把 source IP, source port, destination IP, destination port 寫進 packet 中，
    收到 packet 時:
      Sender: [source IP, source port, destination IP, destination port, acknowledge #,]
      Receiver: [source IP, source port, destination IP, destination port, sequence #, data,]
      Agent: [destination IP, destination port, X, X, data or ack #, ...]
    Agent 會解讀 packet 中的資訊來決定 forward 對象，Receiver 會解讀 packet 決定回傳對象。

Q: Buffer 要怎麼照順序存，什麼時候要傳下一輪的 packets，哪些 packets 是重傳的，要怎麼知道 buffer 已經滿了......?
A: 我加了很多 flags 去判斷不同形況，順便把相近的東西包成 classes 讓 code 變得乾淨一點:)

Q: 我這次沒有用 C 寫作業跟大家的寫法不一樣好痛苦 Q____Q !!!!!
    有人問我怎麼處理時我還要把方法翻譯成 C 好痛苦 Q____Q !!!!!
    Dart 可以用的 library 好少網路上資料好少好痛苦 Q____Q !!!!!
    最後壓縮出來的檔案大小實在有夠大的真的好痛苦 Q____Q !!!!!
A: 寫完之後就不痛苦了，我學到了好多新東西 :D

這是我第一次用 OOP 語言寫 command-line application，碰到許多以前寫 web app 沒用過的東西，code 的架構也與 C 相差許多，不過我意外地找到了這兩種不同語言的關聯性，也算是學了兩遍 socket programming :D (希望助教在看我的 code 時不會太吃力......)