2個重要用

ex: ssh

TCP interative都是短封包

41byte封包很常見(TCP-20B, IP-20B)

*2008/12/10*

*2008/12/10*



**Chapter 19:**

**TCP Interactive Data Flow**

*Network & System Lab, NSYSU*

*1*



**Introduction**

* **Examining the transfer of interactive data using TCP**
* **Using delayed acknowledges**
* **Using Nagle algorithm**

*Network & System Lab, NSYSU*

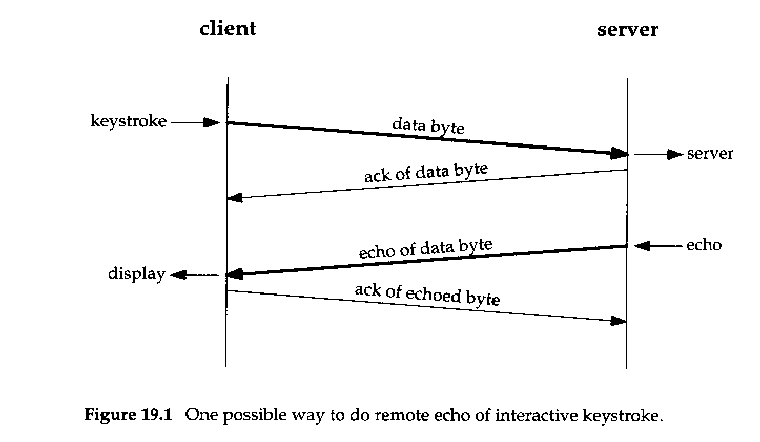
*2*

合併2個ACK

減少一次傳輸

*2008/12/10*

*2008/12/10*



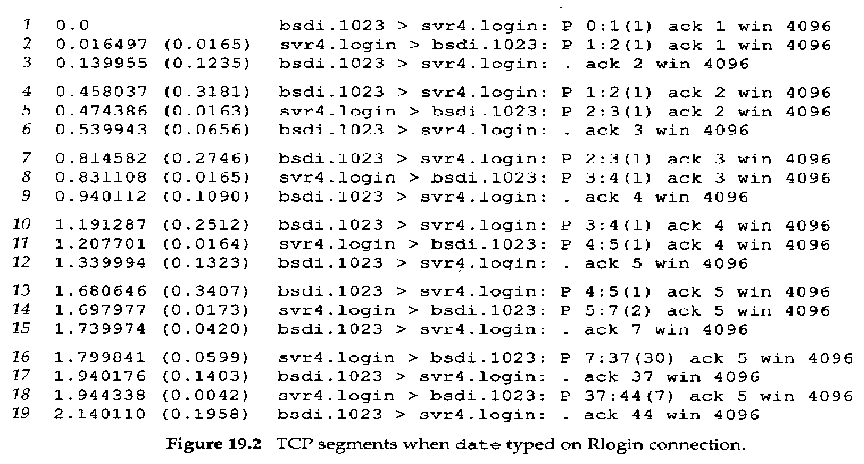
**Interactive Input**

* **Examining interactive data on Rlogin connections:**

Normally being combined

*Network & System Lab, NSYSU*

*3*



**Interactive Input (Cont.)**

* **A Rlogin Example:**

*Network & System Lab, NSYSU*

*4*

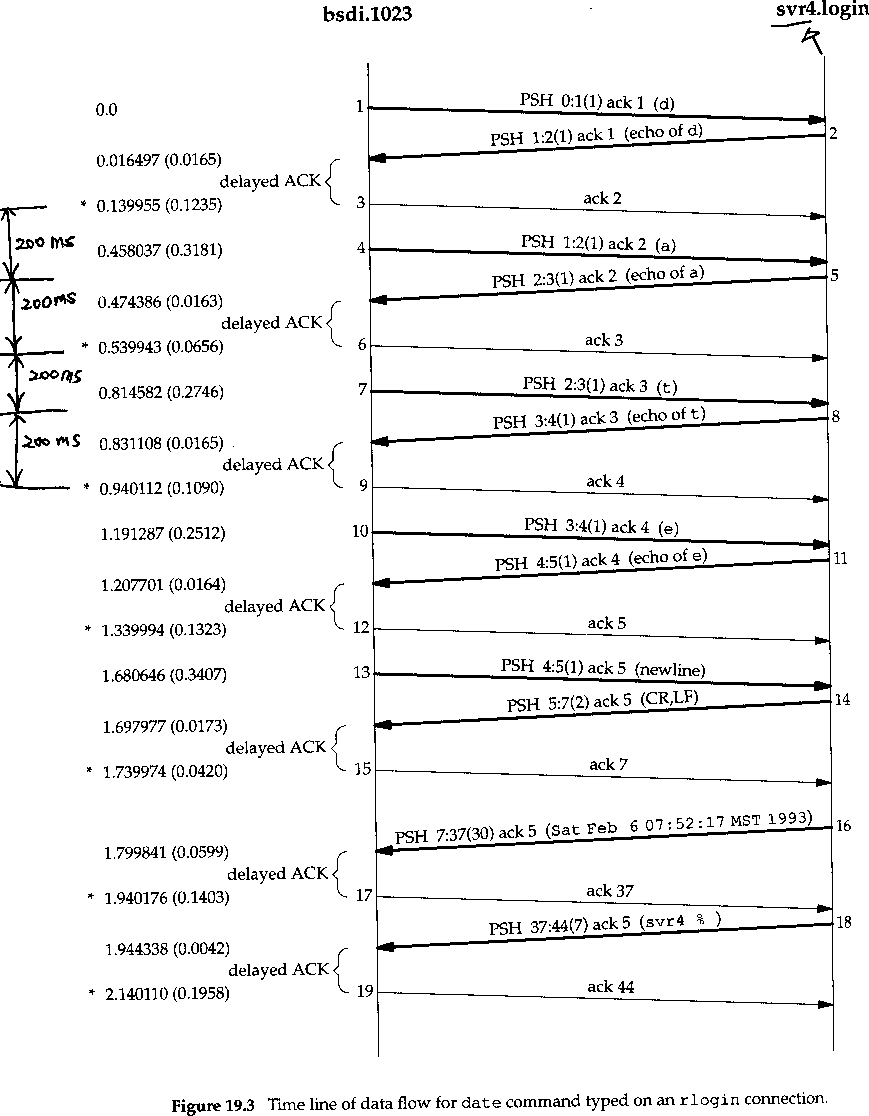
實務上會delay0.2s

piggyback: 揹著

不馬上送ack

*2008/12/10*

*2008/12/10*



**Delayed Acknowledge**

•

2

1

3

*Network & System Lab, NSYSU*

*5*



**Delayed Acknowledge (Cont.)**

* **Delayed ACKs:**
  + Normally TCP does not send an ACK the instant it receives data. Instead, it delays the ACK, hoping to have data going in the same direction as the ACK, so the ACK can be sent along with the data. This is sometimes called having the ACK *piggyback* with the data.
  + Most implementations use a 200-ms delay -- that is, TCP will delay an ACK up to 200ms to see if there is data to send with the ACK.
  + Every 200 ms relative to when the kernel was bootstrapped, NOT relative to when the data received.

*Network & System Lab, NSYSU*

*6*

*2008/12/10*

*2008/12/10*



**Nagle Algorithm**

* **Small packets called *tinygram*: 41-bytes packets:**
  + 20 : IP header
  + 20: TCP header
  + 1: data
* **The effects of tinygrams:**
  + normally not a problem on LANs
  + add to congestion on wide area networks

*Network & System Lab, NSYSU*

*7*



**Nagle Algorithm (Cont.)**

* **A simple and elegant solution of tinygrams: Nagle algorithm:**
  + When a TCP connection has outstanding data that has not yet been acknowledged, small segments cannot be sent until the outstanding data is acknowledged. (i.e., window size = 1)
* **The beauty of this algorithm: self-clocking**
  + the faster the ACKs come back, the faster the data is sent

*Network & System Lab, NSYSU*

*8*

Nagle algo

1. 合併一定數量的輸出資料後一次送出。
2. 只要有已送出的封包尚未ack，傳送者會持續緩衝封包，直到收到ack的資料才送出。

self-colcking: 中央標準時間

Nagle Algorithm具有self-clocking效果(靠ack回來的速度決定)

考

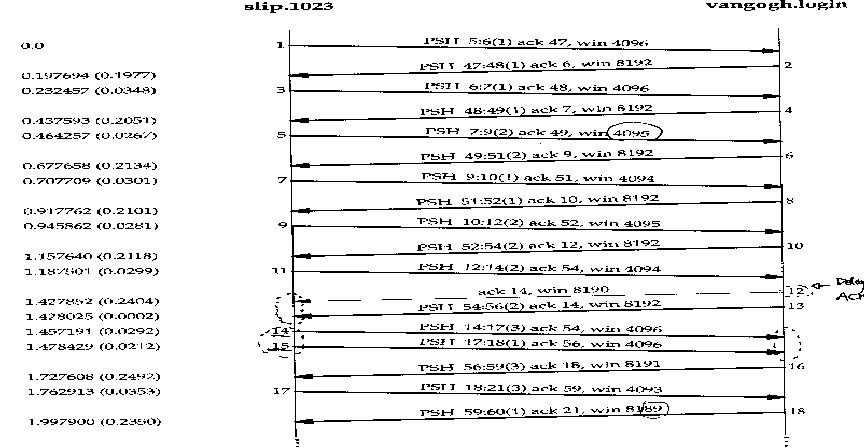
outstanding data: 送出去，但還沒有ack回來

通常區網沒有塞車的問題

小封包

*2008/12/10*

*2008/12/10*



**Nagle Algorithm (Cont.)**

* **Examination from slip to vangogh.cs.berkeley:**

Delayed

ACK

*Network & System Lab, NSYSU*

*9*



**Nagle Algorithm (Cont.)**

* **Things we notice:**
  + the lack of delayed ACKs from slip to vangogh: on this example there is always data ready to send before the delayed ACK timer expires
  + the various amounts of data being sent from the left to the right: by the Nagle algorithm
  + the 14th, 15th segments response the 12th, 13th segments, that’s also obey the Nagle algorithm
  + the 12th segment is a delayed ACK, for without data
  + the 18th segment echoed only 1 byte: this was done by the TCP module in the kernel, and the data not up to the application of the server yet; the advertised window showed 8189 (not 8191) also indicated the same thing

*Network & System Lab, NSYSU*

*10*

*2008/12/10*

*2008/12/10*

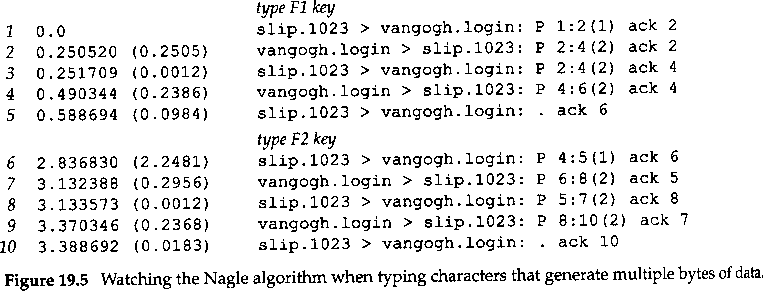


**Disable the Nagle Algorithm**

* **There are times when the Nagle algorithm needs to be turned off:**
  + the X Window System server: small messages (mouse movements) must be delivered without delay to provide real-time feedback for interactive users doing certain operations
  + terminal’s special function keys

*Network & System Lab, NSYSU*

*11*



**An Example (with the Nagle Algorithm)**

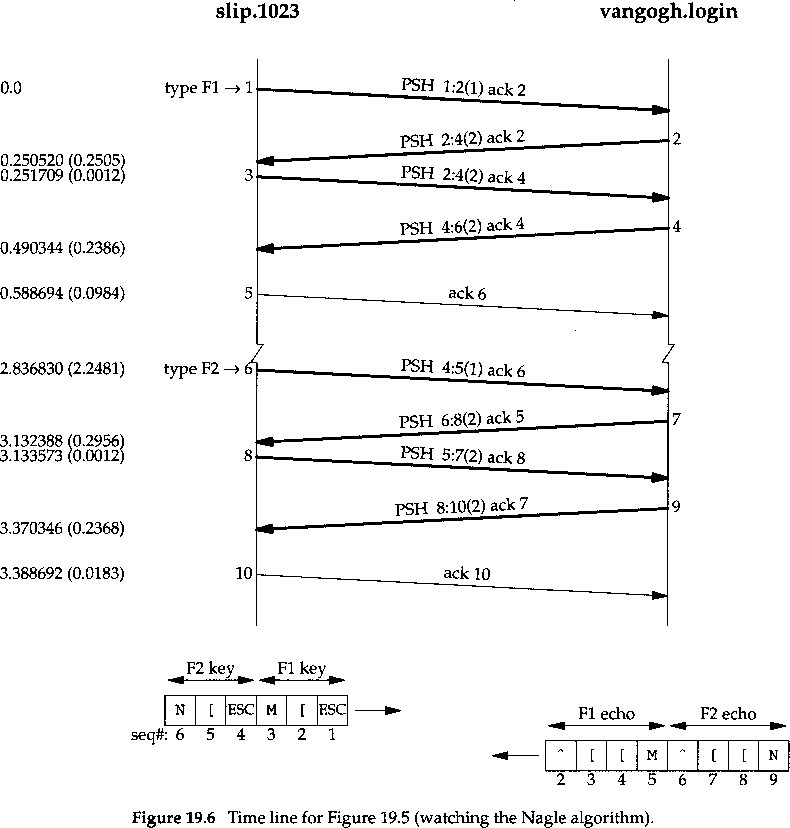
* **From Slip to vangogh.cs.berkeley.edu**
* **type the F1 function key and via versa..**

*Network & System Lab, NSYSU*

*12*

*2008/12/10*

*2008/12/10*

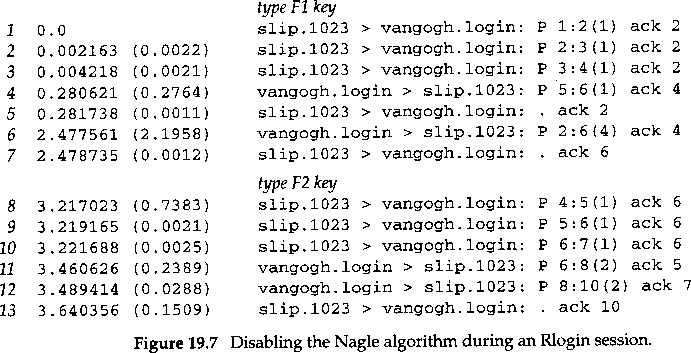


**An Example (with the Nagle Algorithm)**

* **(continued…)**

*Network & System Lab, NSYSU*

*13*



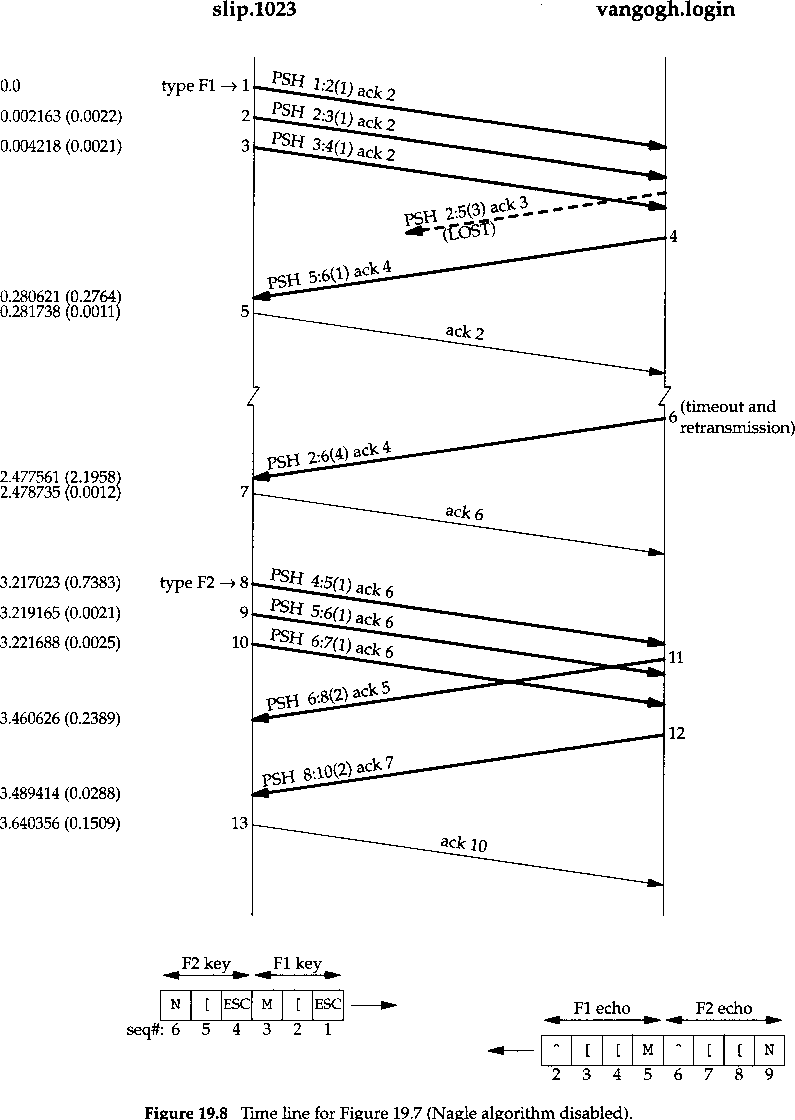
**An Example (without the Nagle algorithm)**

* **The same command typing as previous one (but disable the Nagle algorithm):**

*Network & System Lab, NSYSU*

*14*

*2008/12/10*



**An Example (without the Nagle algorithm)**

* **(continued..)**

Repacketization: retransmission with the lost packets

*Network & System Lab, NSYSU*

*15*



**Window Size Advertisements**

* **In figure 19.4:**
  + slip advertises a window of 4096
  + vangogh advertises a windows of 8192
  + Segment 5 advertises a window of 4095 bytes => still 1 byte in the TCP buffer for the application ( the Rlogin client) to read
  + The server normally advertises a window of 8192 bytes, because the server’s TCP has nothing to send until the Rlogin server reads the received data and echoes it. The data from the server is sent after the Rlogin server has read its input from the client.
  + The client TCP, on the other hand, often has data to send when the ACK arrives, since it’s buffering the received characters just waiting for the ACK. When the client TCP sends the buffered data, the Rlogin client has not had a chance to read the data received from the server, so the client’s advertised window is less than 4096.

*Network & System Lab, NSYSU 16*

*2008/12/10*

*2008/12/10*



**Summary**

* **Interactive data**
* **Tinygrams**
* **Delayed acknowledge (piggyback)**
* **Nagle algorithm**

*Network & System Lab, NSYSU*

*17*