Assignment 3

1. This question concerns TCP congestion control algorithms. The followings are assumed:
    - ◊ Use the slow start, congestion avoidance, fast retransmit, and fast recovery algorithms specified in the standard.
    - ◊ *ssthresh* is set to *max{Fightsize/2, 2MSS}* when retransmission takes place.
    - ◊ The initial *ssthresh* is assumed to be very large. The initial RTO is 2 seconds.
    - ◊ For clarity, *cwnd* is expressed in terms of the number of MSS-sized segments.
    - ◊ The sender begins the congestion avoidance phase when *cwnd* ≥ *ssthresh*.
    - ◊ The sender buffer is always full.
    - ◊ The receiver's advertised window is always large, and then the sender's offered window is solely determined by *cwnd*.
    - ◊ Nagle's algorithm is on. The receiver acknowledges every other MSS-sized segment received.

a) Despite the usefulness of the fast retransmit algorithm based on the third duplicate ACK, a TCP sender may not always be able to use the algorithm to speed up retransmissions. One example is illustrated in Figure 1. Explain why the fast retransmit algorithm cannot be used in this example.

<1>、 Fast retrainsmit:

流程：当收到第三个重复的ack：

Set ssthresh to no more than max{FlightSize/2, 2mss}.

Retrainsmit the missing segment.

在0.5时刻，cwnd=3.

发出3个包后，此时 Effective Wnd=0

Naggle算法阻止发送方发送更多的数据.

第3个packet丢失，只能收到至多两个重复的ack.

无法达到fast retrainsmit算法的启动条件

只能等到1个RTO发生超时后重传.

b) Please describe the events that happened at 2.5 seconds and the changes of *ssthresh* and *RTO*.

(b). 为了保证数据的可靠传输，发送端有一段时间内（RTO）没有收到ACK报文，就会对每个segment进行重传.

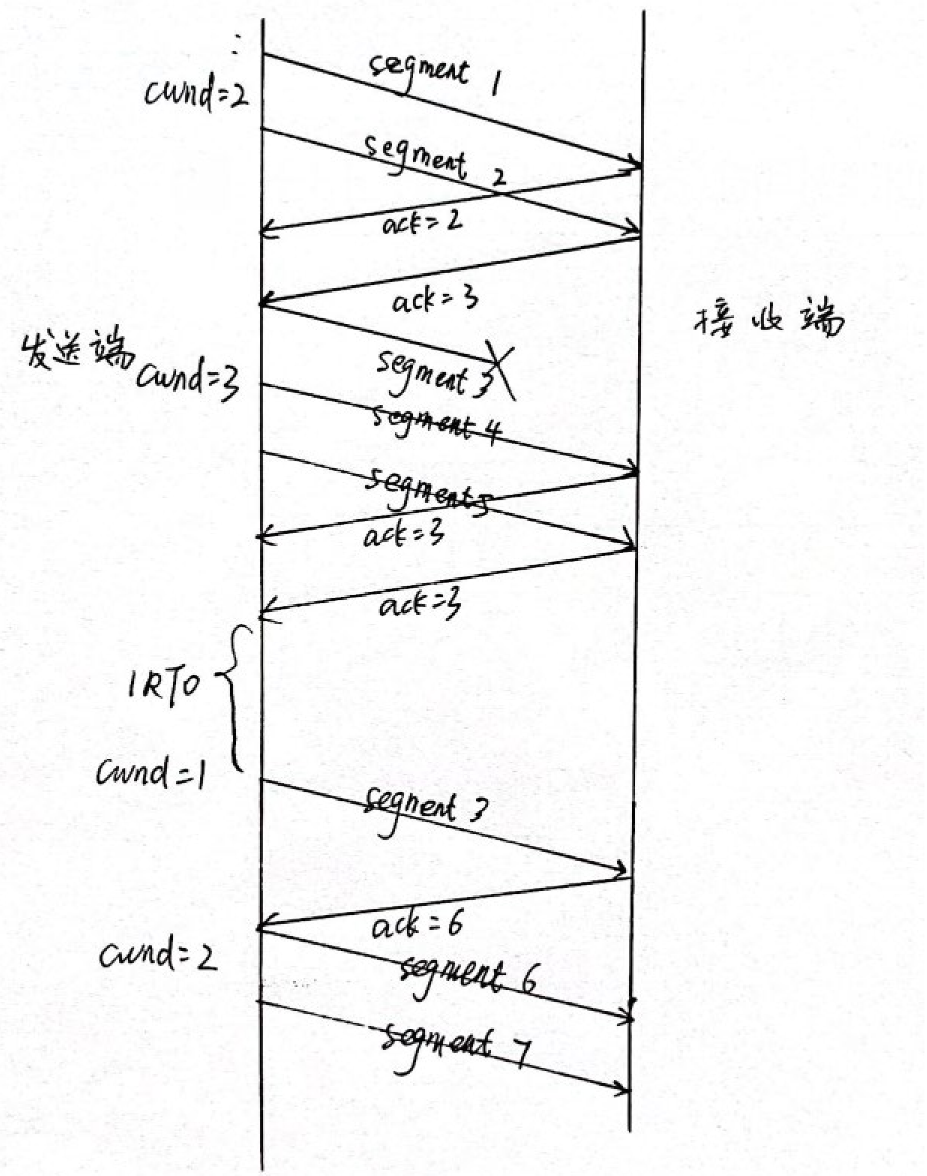
此时 ssthresh 被设置为 2MSS

cwnd 设置为 1个 MSS（慢启动）

RTO 进行 exponential backoff.

c) Please describe the events happened in 3 seconds. Why were new segments sent at this time?

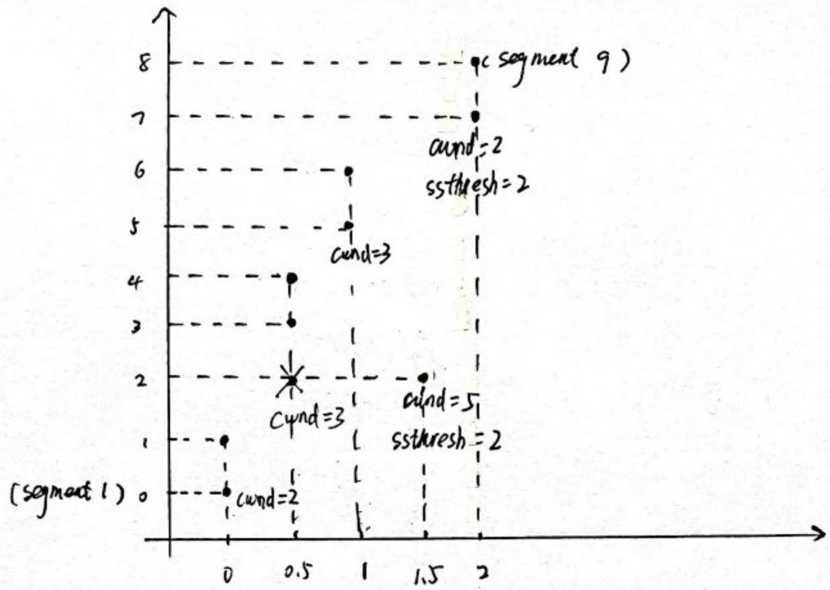
(c)、



d) To solve the problem exhibited in part (a), some researchers have proposed a

Limited Transmit algorithm (RFC 3042), which allows a TCP sender to send new data segments after receiving the first and second duplicate ACKs. However, the value of *cwnd* is unchanged. In other words, the algorithm allows the sender to transmit two segments beyond what *cwnd* allows upon receiving duplicate ACKs. Moreover, the fast retransmit and fast recovery algorithms still apply when the third duplicate ACK is received. Draw a diagram similar to Figure 1 when the Limited Transmit algorithm is used (up to and including the sending of segment 9). Discuss whether the Limited Transmit algorithm improves the TCP throughput performance.

<d>.



发送至 segment 9 (packet number =8)

Figure 1 用时 3.5 Time

Limited transmit algorithm 用时 2 Time.

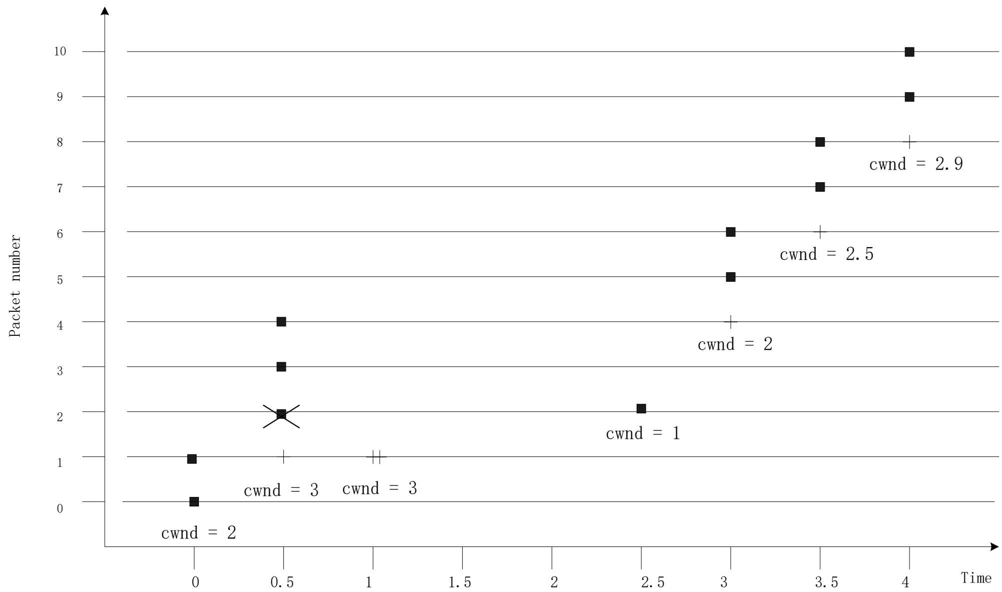
在该场景下提高了TCP的吞吐量.

但是在其他拥塞引发的丢包问题中,

使用Limited transmit algorithm 会增大网络上数据包的量.

有可能加重网络负担

其它场景下是否能提高TCP吞吐量需实验判断。



Legends: ■ TCP data segments sent by the sender

\+ ACKs received by the sender (for clarity, an ACK's value refers to the largest number of the segments received, not the number of the next expected segment)

× Segments dropped in the network

Figure 1

2. Suppose you have a Host *C*, a local name server *L*, and authoritative name servers A_{root}, A_{com}, and $A_{google.com}$, where the naming convention A_x means that the name server knows about the name zone *x*. A_x is a variable and NOT a hostname. A_{root} is a root name server known to *L*, with IP address 198.41.0.4. Assume that all name servers initially have nothing in their caches.

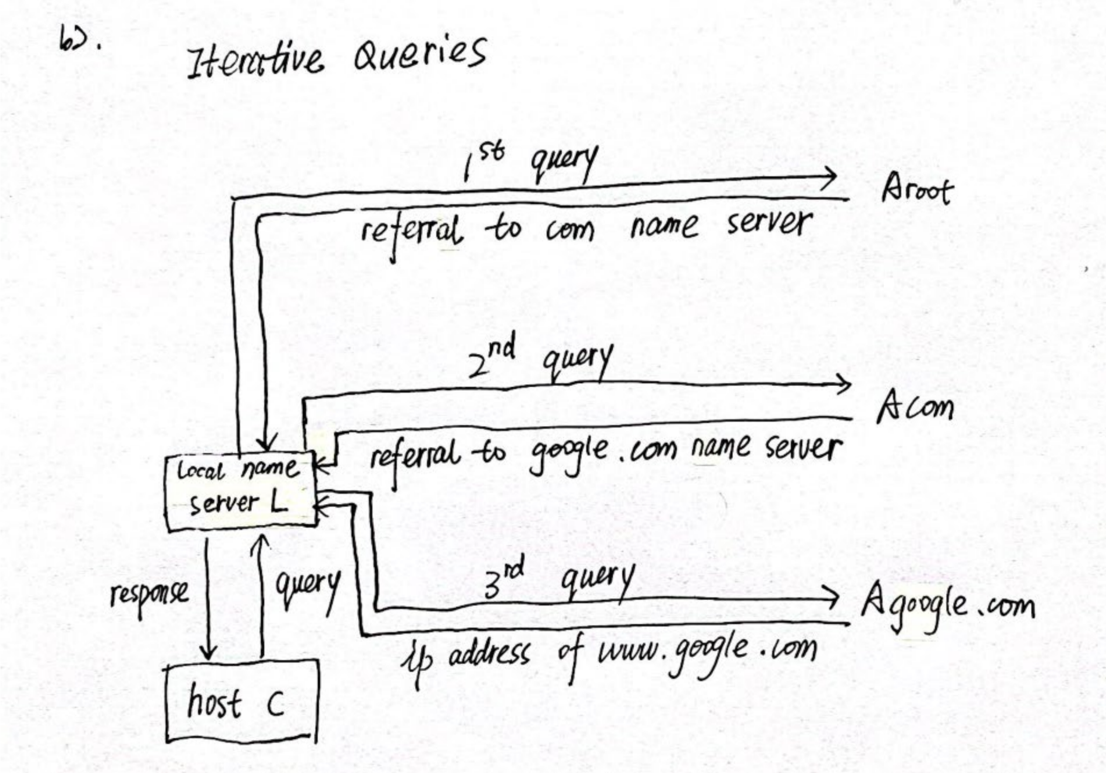
a) Using the resource records below, provide the hostnames and IP addresses for A_{com} and $A_{google.com}$.

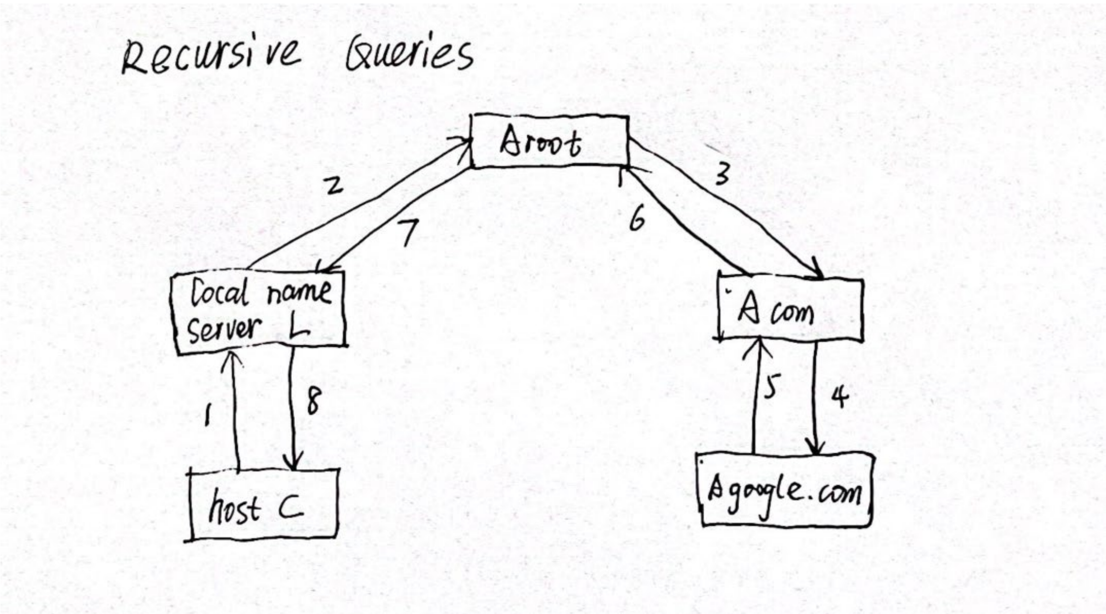
Name Server Variable	Resource Record
A_{root}	{com, a.gtld-servers.net, NS, IN}
A_{root}	{a.gtld-servers.net, 192.5.6.30, A, IN}
A_{com}	{google.com, ns1.google.com, NS, IN}
A_{com}	{ns1.google.com, 216.239.32.10, A, IN}
$A_{google.com}$	{www.google.com, 66.102.7.104, A, IN}
$A_{google.com}$	{mail.google.com, 66.102.7.83, A, IN}

a) A com : a.gtld-servers.net 192.5.6.30

A google : ns1.google.com 216.239.32.10

b) List the sequence of DNS queries and corresponding resource records exchanged when *C* wants to lookup the address for www.google.com.

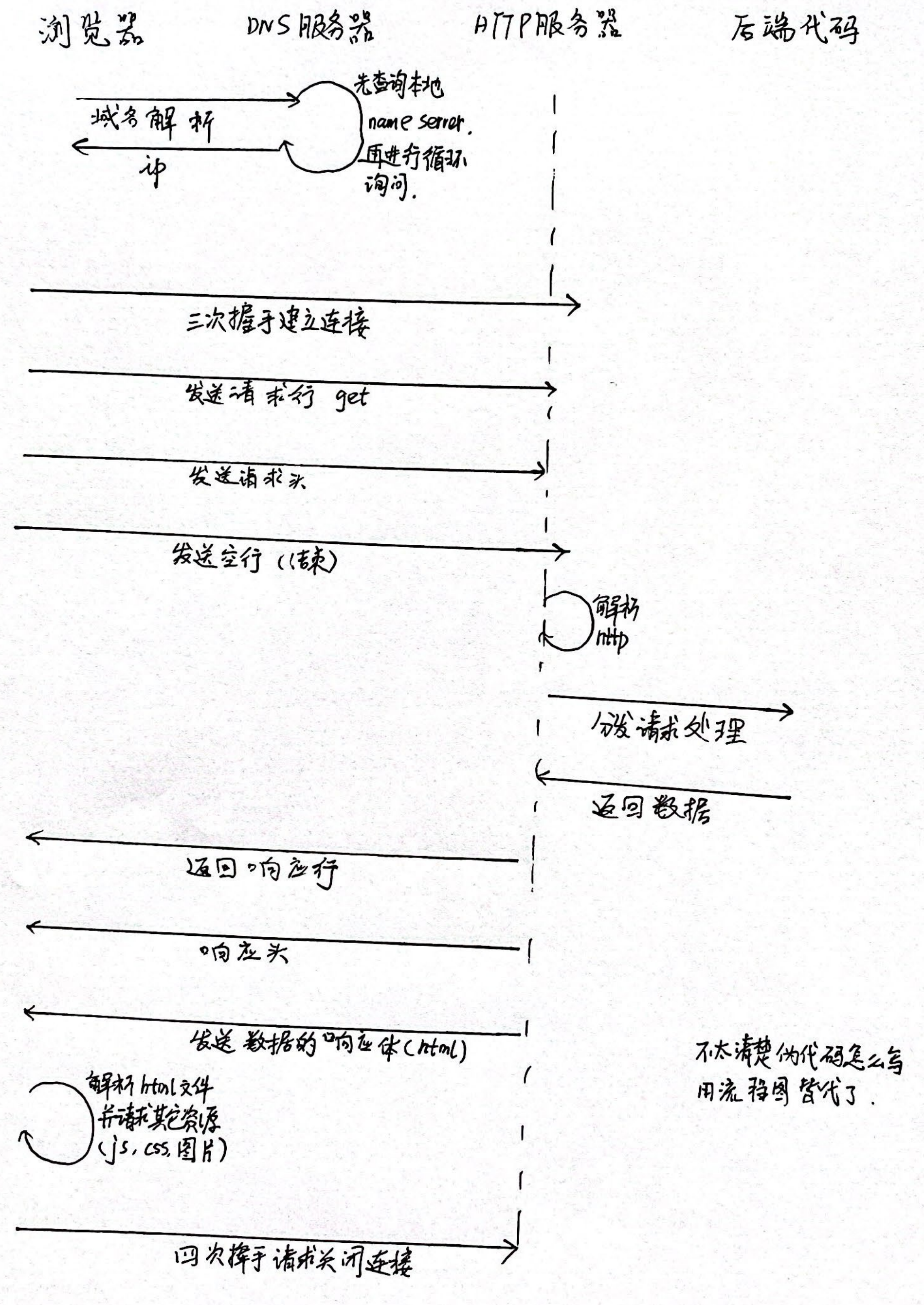




3. Please answer the following questions pertain to the usage of HTTP protocol.

a) Describe the operation of a Web server with high-level pseudocode. It is sufficient to show operation for only the case of HTTP GET.

3、(a)



b) If the use of conditional GET requires that a proxy server should always contact the origin server for every object in its cache to check if the object has been modified, then where is the savings in downloading time achieved for web pages? Briefly explain in a couple of sentences.

<b>. Web代理服务器缓存会保存ISP客户访问过的服务器的Web页面副本，如果缓存的页面与原始服务器页面一致，则ISP的所有客户在访问该页的时候只需从代理服务器拉取副本，提高了访问服务器Web页效率。

c) Suppose that the file size of an object in a HTTP server is 24,000 bytes. The object is cached on the proxy server after the first access to the object. Suppose on each subsequent access to the object by a client, the proxy server finds that the object is not modified 90% of the time. Suppose a conditional GET request when the file is not modified requires only 200 bytes of message exchange. Compute the overall savings in the percentage of the data to be downloaded by using conditional GET as a function of parameters specified above for *x* requests for the object from the clients, as opposed to using the normal access without using conditional GET.

<c>、

$$\frac{24000(1-90\%)x + 200 \cdot 90\% x}{24000x} = 10.75\%$$

4. Consider the following protocol for *A* to authenticate *B* using public-key signature. N_A is a nonce selected by *A*, and $SIGB(N_A)$ is B's signature over N_A.
   $(1)\ A \rightarrow B : N_A$ $(2)\ B \rightarrow A : SIGB(N_A)$
   Like many other protocols that you have seen before, this one suffers from an impersonation attack. Consider that *C* impersonates *B* by creating two sessions: one with *A* and the other with *B*. Fill in the missing messages below and explain your answer. The notation C_B refers to *C* claiming to be *B*.
   $(1)\ A \rightarrow C_B : N_A$ $(1')\ C \rightarrow B : ?$ $(2')\ B \rightarrow C : ?$ $(2)\ C_B \rightarrow A : ?$

(1). $A \rightarrow C_B : N_A$

A使用随机数N_A向B索取公钥，却被C截获。

(1') $C \rightarrow B : N_A$

C将消息转发给B，B无法分辨这条消息是否真的从A那里发来。

(2') $B \to C$: $SIGB(N_A)$

B回应A的消息，并附上其签名的公钥

(2). $C_0 \to A$: $SIGC(N_A)$

C用自己签名的公钥替代了B签名的公钥，并转发给A.

之后C再次截获到A发送给B的消息时，可以使用C的私钥对其解密和篡改，然后使用B的 $SIGB(N_A)$ 中的公钥对消息进行再次加密，当B收到消息时，会相信这是从A来的

5. Assume A & B can hear each other, B & C can hear each other, and C & D can hear each other. No other nodes can hear each other. For parts (a) and (b) also assume that RTS and CTS are not being used.

a) If C wants to send to B while A is sending to B, will a collision occur? Why or why not? Will this be considered hidden or exposed terminal?

(1). 会. 基站A向基站B发送信息，由于基站C未侦测到A在向B发送，故A和C同时将信号发送至B，引起信号冲突。这是隐藏终端问题。

b) If B wants to send to A while C is sending to D, will a collision occur? Why or why not?

(2). 会. B在发送节点C的范围内，而在接收节点D的范围外，此时B是暴露终端，会因监听到C的发送而延迟发送，但实际在接收节点的范围外，它的发送不会造成冲突.

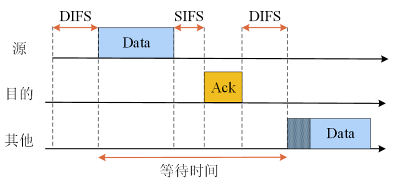
c) How does RTS/CTS overcome the problem of hidden or exposed terminal?

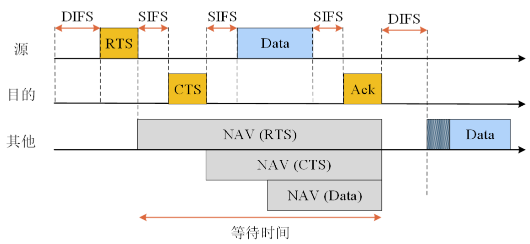
(3)、在单信道条件下使用RTS/CTS的方法只能解决隐藏发送终端问题，无法解决隐藏接收终端和暴露终端问题。只有采用双信道方法，即用控制信道收发控制信号，用数据信道收发数据，才可以解决全部问题。

对于隐藏发送终端问题，当A向B发送数据时，先发送一个控制报文RTS（request to send），B接收到RTS后，会回应CTS（clear to send）控制报文，A收到CTS后才会向B发送报文，如果A没有收到CTS，A认为发生了冲突，重发RTS，这样隐藏发送终端C能侦听到B的CTS，知道A将向B发送报文，C延迟发送，解决了隐藏发送终端的问题，最后，B接收完数据后，向所有基站广播ACK，这样，其它的基站再次平等竞争信道。

对于暴露终端问题，当B向A发送数据时，C只听到RTS控制报文，知道自己是暴露终端，认为自己可以向D发送数据，C向D发送控制报文RTS，如果是单信道，来自D的RTS会与B发送的报文数据冲突，C和D无法成功握手，不能向D发送报文。

d) If the packets being sent by the wireless nodes are very short (few bytes each), would RTS/CTS be a useful mechanism if link--layer acks are already being used? Why or why not?





(d). 由于短帧冲突信道浪费少，而使用RTS/CTS需要多等待一个CTS报文和2个SIFS的传输间隙，所以不能明显改善效率。