Fair Queuing

idea: 哪个pkt的完成传输的时间最短，就先传哪个pkt.

具体解释：一旦决定传哪个pkt，就要把整个pkt传完.

假设此时正在router上传 pkt A

预计pkt B将在 8:00 被传完.

- 若新pkt (pkt B) 在 8:20 (pkt A被传完之后) 到达router

  则 time when the router finishes transmitting pkt B

  即 $F_i = \max(F_{i-1}, A_i) + P_i = 8:30$

  F_{i-1} = 8:00, A_i = 8:20, $\max(F_{i-1}, A_i)$ = 8:20

  P_i: 传pkt B需10 min

- 若新pkt (pkt B) 在 7:30 (pkt A被传完之前) 到达 router

  则 $F_i = \max(F_{i-1}, A_i) + P_i = 8:10$

  F_{i-1} = 8:00, A_i = 7:30, P_i: 10 min.

  $\max(F_{i-1}, A_i)$ = 8:00

rule 1:

每一个新pkt arrive，计算它的 F_i，作为its timestamp.

the next pkt to transmit is always the pkt that has lowest timestamp (即 it should finish transmission before all others)

rule 2:

每传完一个pkt，作一次决定：传哪个pkt接下来.

当有pkt正在被传时，不管新来的pkt size是大是小，都不能中断正在被传的pkt.

Weighted Fair Queuing

assign weight to each flow (queue)

weight → how many bits to transmit each time the router services that queue.
- control the percentage of the link's bandwidth that flow will get
- higher weight get more bandwidth.

Simple Fair Queuing:

if flow 1 is active	weight	佔 bandwidth
flow 1	1	1/3
2	1	1/3
3	1	1/3

one bit is transmitted per round, fair in each flow

Weighted Fair Queuing

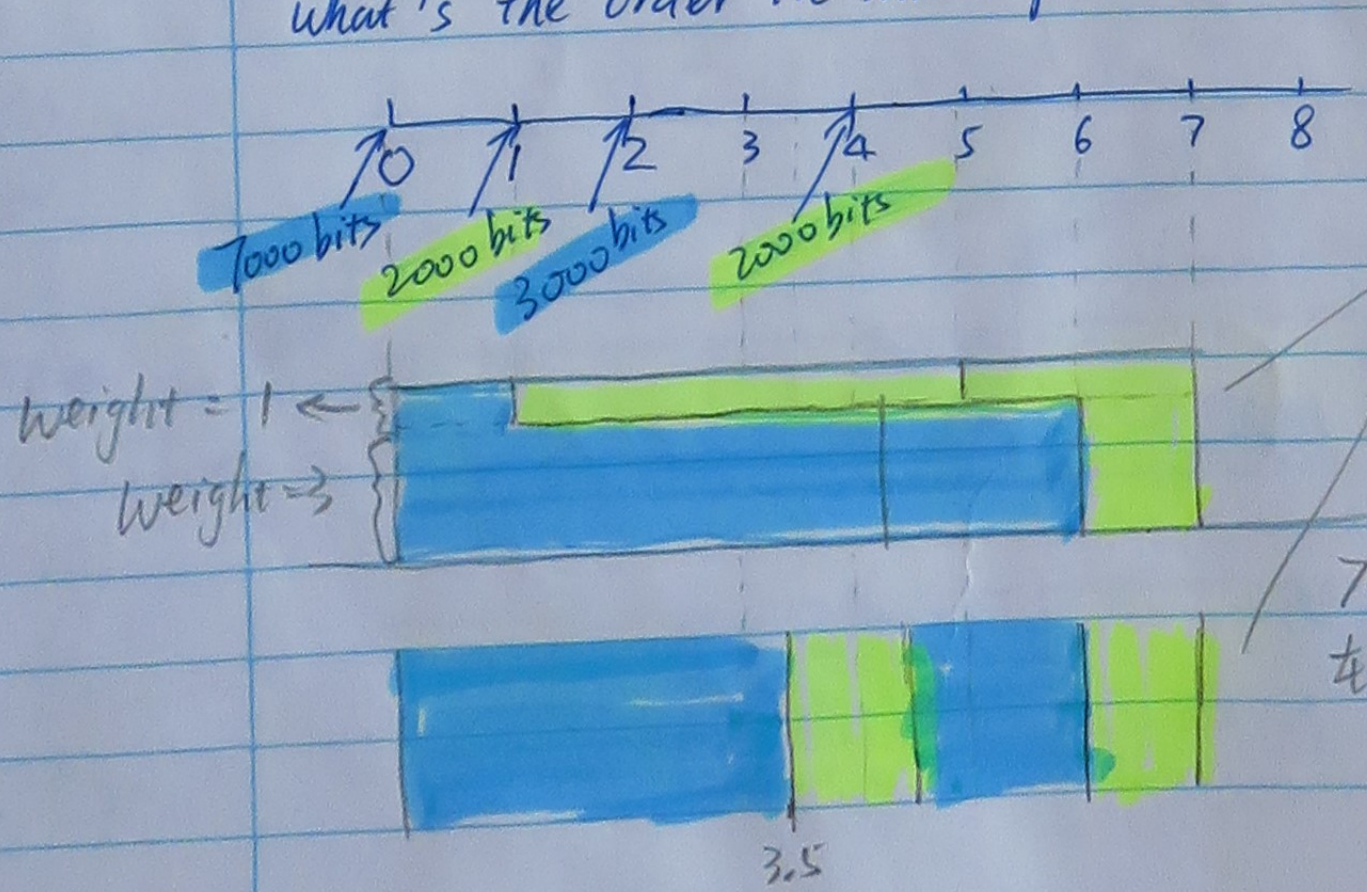
if flow 1 is active	weight	佔 bandwidth
flow 1	2	1/3
2	1	1/6
3	3	1/2

in each round → 2 bits from flow 1 is transmitted
1 ... 2 ...
3 ... 3 ...

$$F_i = \max(Finish_{i-1}, Arrive_i) + \frac{\text{Packet size of } i}{\text{weight of } i\text{'s queue}}$$

佔 yellow: weight = 1
blue: 3

Output
what's the order in which pkts leave the router? blue, yellow, blue yellow.



leave router order:
藍黃藍黃

$7000 \div (2 \times 10^6) = 3.5$ ms
在 3.5ms, 收到 黃 2000 bits
藍 3000 bits

若传黄2000bits $Finish_{黄2000} = \underbrace{3.5\times10^{-3}}_{sec} + \frac{2000\div(2\times10^6)}{3} = 3.8\times10^{-3}$ 小

$Finish_{黑3000} = 3.5\times \quad + \frac{3000\div(2\times10^6)}{3} = 4\times10^{-3}$ 大

先传黄2000

chp6 exe10 同时到达

weight	flow	size: 100	100	100	100
1	flow 1	4	3	2	1
4	2	200		190	
		6		5	
1	3	50		110	
		8		7	

fair queuing

pkt	size	Flow	F_i
1	100	1	100
2	100	1	200
3	100	1	300
4	100	1	400
5	190	2	190
6	200	2	390
7	110	3	110
8	50	3	160

$A_1 = 0$

accumulative F_i

$F_i = F_{i-1} + P_i$

根据 F_i 排序： $100 < 110 < 160 < 190 < 200 < 300 < 390 < 400$

⇒得 pkt 的 transmit 顺序： 1 7 8 5 2 3 6 4

先 最后

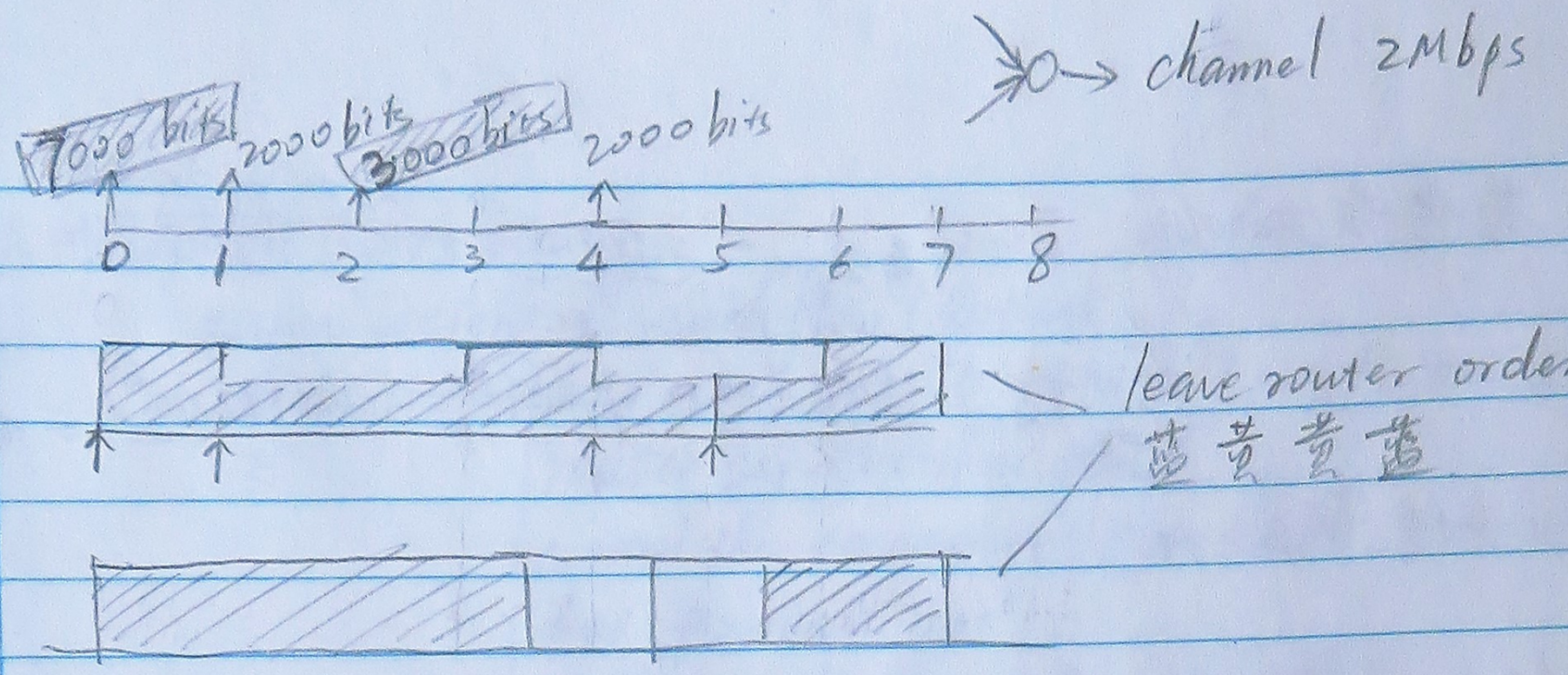
weighted fair queuing

∵ flow 2 has weight 4 ⇒ $F_i = F_{i-1} + \frac{P_i}{4}$

	pkt	F_i
weight=1	1	100
	2	200
	3	300
	4	400
weight=4	5	$190\div4 = 47.5$
	6	$47.5 + \frac{200}{4} = 97.5$
weight=1	7	110
	8	160

F_i 从小到大： $47.5 < 97.5 < 100 < 110 < 160 < 200 < 300 < 400$

pkt 从先到后传： 5 6 1 7 8 2 3 4



新 pkt 到达和时间

↙ 在新 pkt 到达时，这一 queue 已经传了多少 bits.

Blue	(ms) (time)		pkt size		
i	t_i	A_i	P_i	S_i	F_i
1	0	0	7000	0	7000
2	2	3000	3000	7000	10000

$1ms \times 2Mbit/sec + 1ms \times 0.5 \times 2Mbit/sec$

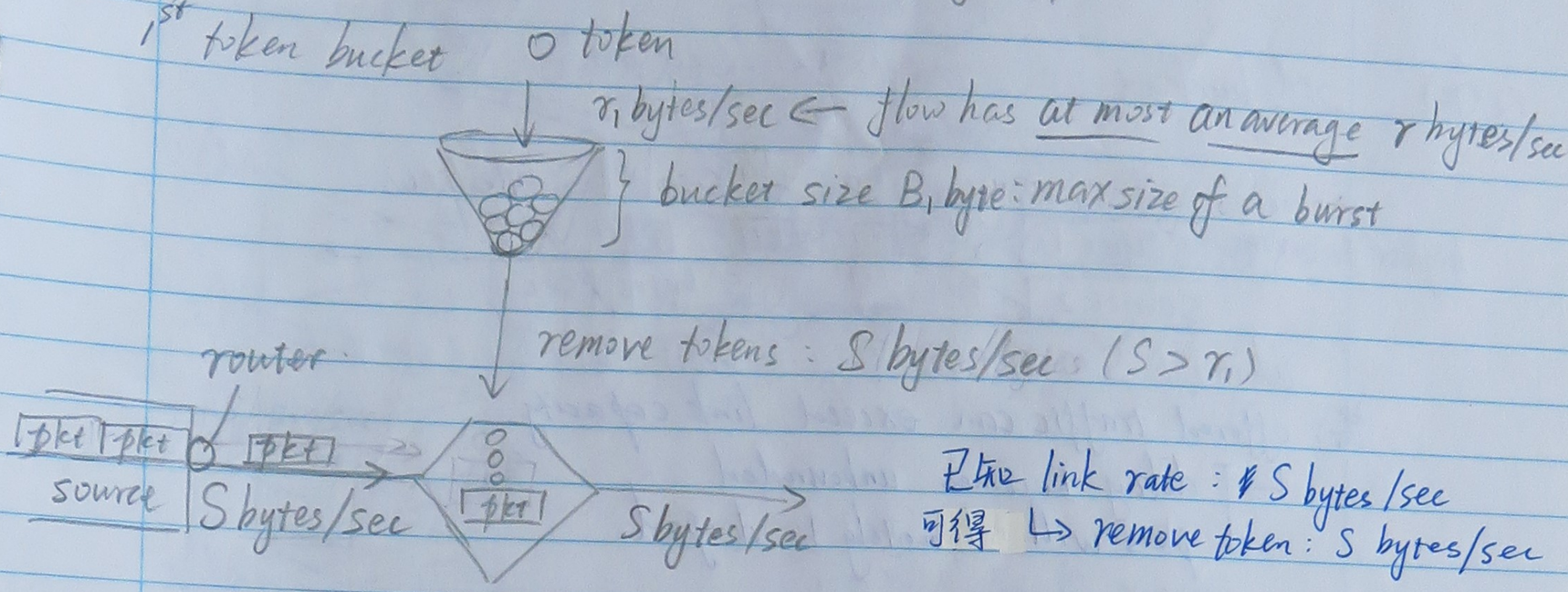
当新 pkt 开始被传时，这一 queue 已经传了多少 bits

当这一 pkt 被传完的那一刻，这一 queue 已经传了多少 bits

Yellow

i	t_i	A_i	P_i	S_i	F_i
1	1	2000	2000	2000	4000
2	4	6000	2000	6000	8000

Q(A) assume all pkts have size : 1 byte/pkt

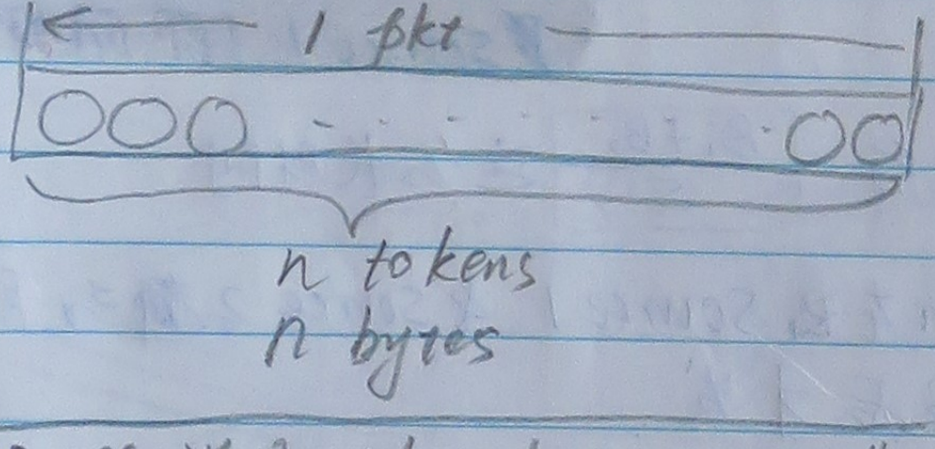


已知 link rate : S bytes/sec
可得 ↳ remove token : S bytes/sec

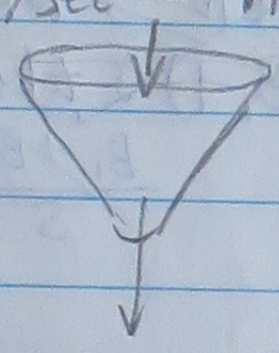
Source sends pkt satisfying a token bucket specification.
What is max delay a pkt experiences (queuing + transmission delay)

$\frac{B_1}{S}$

定义 one token contains 1 byte ∴
- to send a pkt of length n (即 n bytes)
  - ↳ I need n tokens



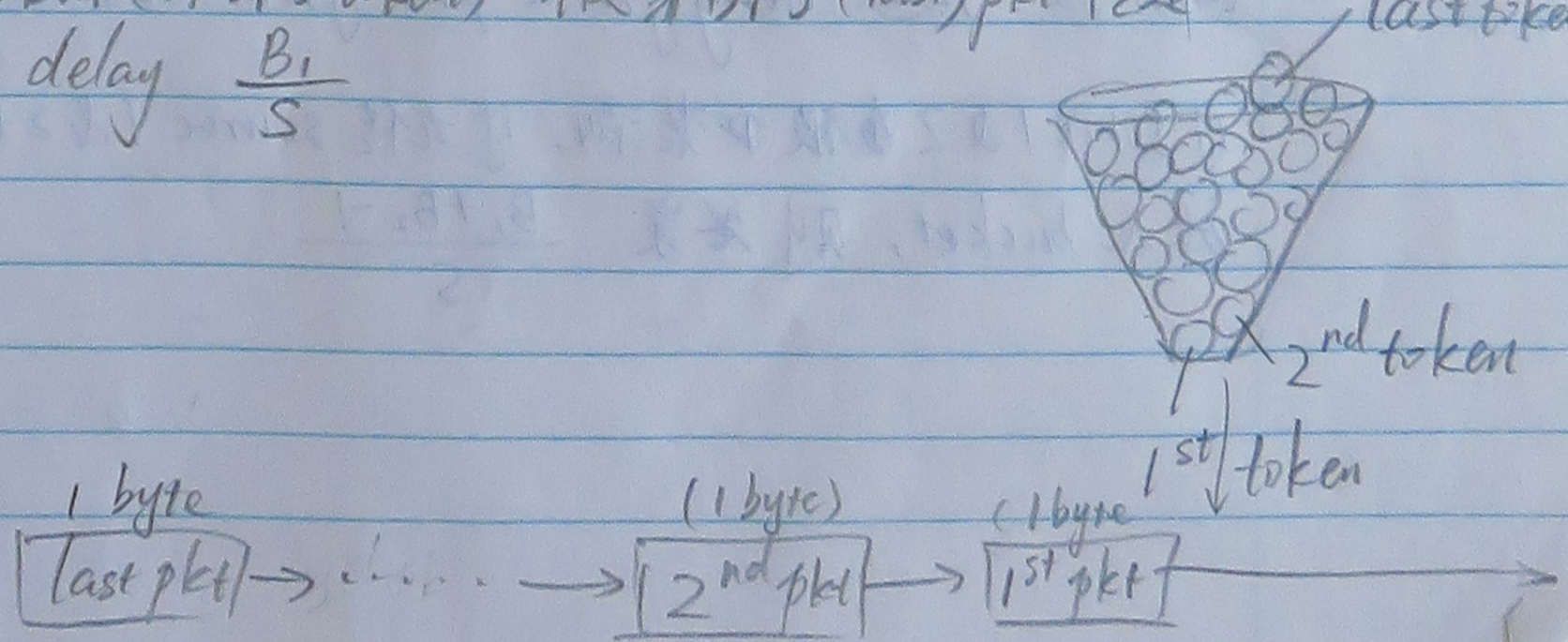
此题中假设
r_1 bytes/sec = r_1 tokens/sec 慢



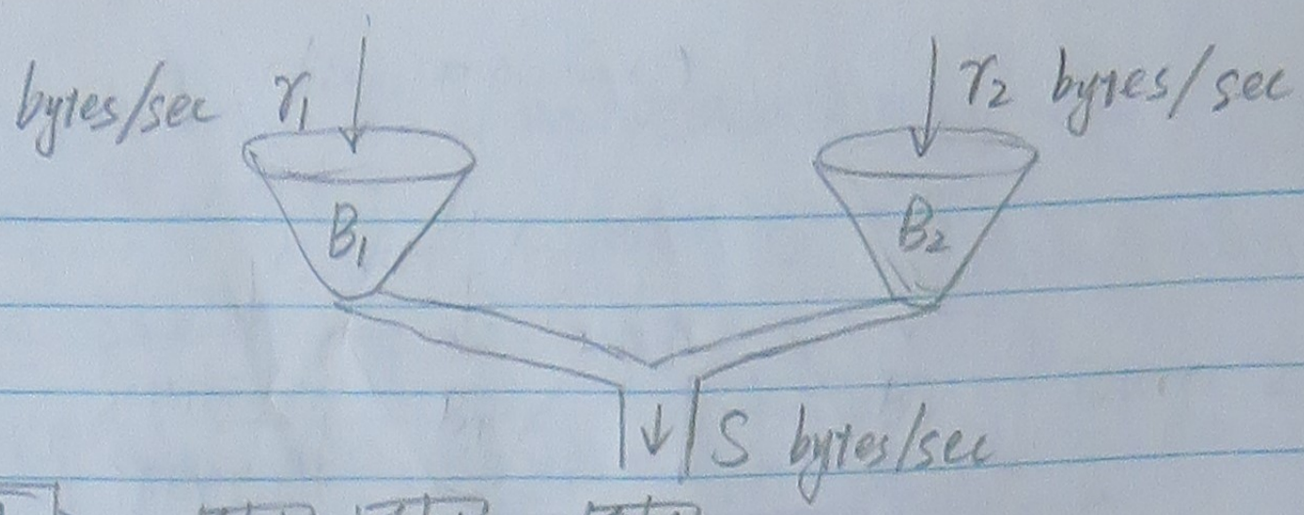
S bytes/sec = S tokens/sec 快.
此题中假设
1 byte/pkt 表示 1 pkt 只装 1 token

$S > r_1$ 说明: token bucket is refilled slower than the link transmits data
r_1 < S

max delay a pkt experience means:
- bucket 被装满了, 共有 B_1 个 token
- 第一个 token (离 bucket 出口最近的 token) is transmitted immediately (给 1st pkt)
  - ... 二 token 被紧接着传走, 给 2nd pkt
  - ⋮
- 最后一个 token (第 B_1 个 token) 被第 B_1 个 (last) pkt 传走
- ∴ max delay $\frac{B_1}{S}$



Q(B) 2nd bucket



what if $S < r_1 + r_2$?

答: offered traffic can exceed link capacity.
↳ [delay becomes unbounded
pkts will ultimately be droped.

Q(C) assume $S > r_1 + r_2$

what is max delay for a pkt from source 1 with FIFO scheduling?

~~worst case~~: 书中定义: 1个 token 装 1个 byte
此题 assume: 1个 pkt 装 1个 byte
1个 pkt 装 1个 token.

worst case 且 { source 1 被装满 ⇒ B_1 个 token 需 B_1 个 pkt
... 2 ... ⇒ B_2 ..... B_2 个 pkt

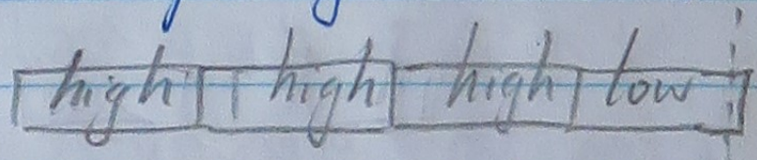
则共需 $B_1 + B_2$ 个 pkt.
假设最后一个 pkt 需装的 token 来自 source 1 (即问题)

则此 pkt 最长需等待 $\frac{B_1 + B_2}{S}$ 这么长时间.

B_1 中的最后一个 token 在 source 1 & source 2 都的所有 token 都被 sent 之后才发.

Q(D) assume — [$S > r_1 + r_2$ (∴ 无积压 backlog)
priority scheduling is used. source 1 has highest priority
what is max delay for a pkt from source 1 ?

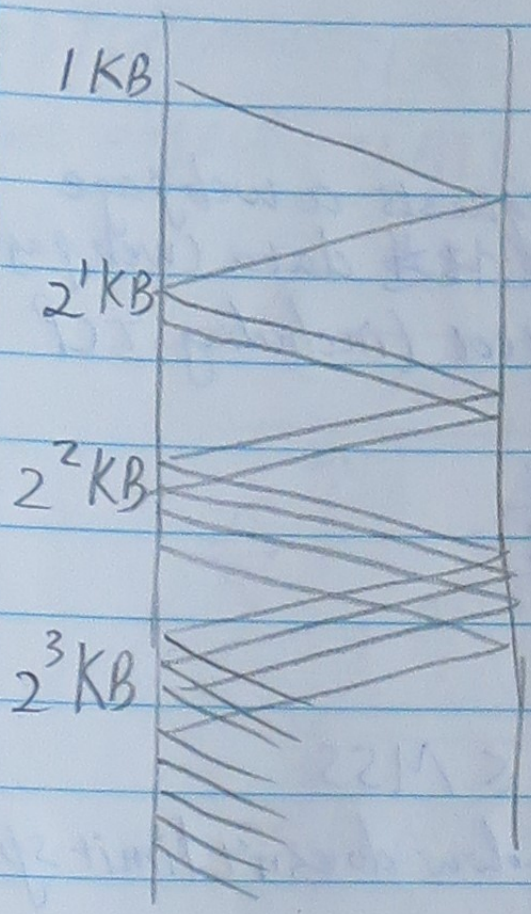
答: $\frac{B_1 + 1}{S}$



Q(E) assume — [$S > r_1 + r_2$
priority schedulling is used, source 1 has highest priority.
what is max delay for a pkt from source 2?

if Sowre 1 & 2 都被全装满, if 在传 source 1 & 2 时, 没有新 token 被放入 bucket, 则答案 $\frac{B_1 + B_2 - 1}{S}$

slow start



Q(A) 已知 { slow start; 1 KB/pkt; no congestion; no lost pkt }

求 要发送 1MB data，需多少RTT？

~~2^n KB = 1 MB~~

2^n KB = 1 MB

2^n KB = 10^3 KB

$n = \log_2(10^3)$

$n = 9.966$

向上取整 $n = 10$

Q(B) a file is 10 MB. 需多少个RTT？

2^n KB = 10 MB

$n = \log_2(10 \times 10^3)$

向上取整 $n = 14$

Q(C) if the time to send file is given by ~~the # of~~ :

# of required RTT × $\frac{\text{link latency}}{\text{已知 50ms}}$

求 effective throughput to transfer the file (10MB)？

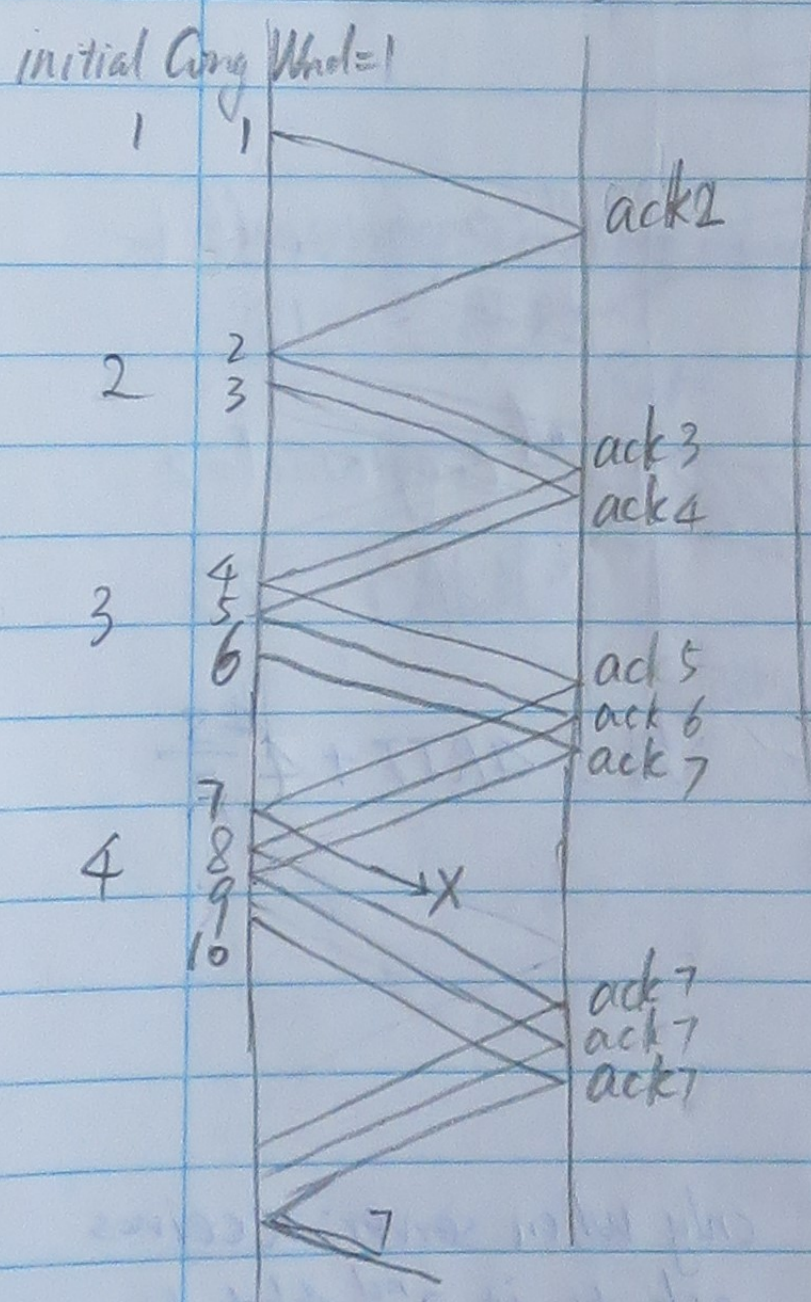
答： $\frac{10\text{MB}}{14 \times 50\text{ms}} = \cdots = 114.3\ \text{Mbps}$

Q(D) what percentage of link bandwidth is used?

已知 1-Gbps link

答： $\frac{114.3 \times 10^6}{10^9} = 0.1143 = 11.43\%$

fast recovery



TCP delay modeling slow start

已知

client link rate R webserver

Max Segment Size = S

Question: how much Time is needed to retrive an object (including TCP connection setup)? in following cases

取: client requests a webpage
server 传送 data (webpage)

(A) if object has a size S 答: $2RTT + \frac{S}{R}$.

1 RTT: set up TCP connection

1 RTT: request object

S/R: transmit object

∵ object size ≤ MSS

∴ congestion window doesn't limit speed.

(B) if object 's size = 3S 答: $3RTT + \frac{3S}{R}$.

~~1 RTT: set up~~

set up TCP connection

request object & get object — $\frac{S}{R}$

request object & get object — $\frac{2S}{R}$

1st cong wnd = 1
数据量: 1S

2nd cong wnd = 2
数据量: 2S

(C) if object 's size = 5S 且 RTT > $\frac{S}{R}$ 答: $4RTT + \frac{4S}{R}$

client server

cong wnd

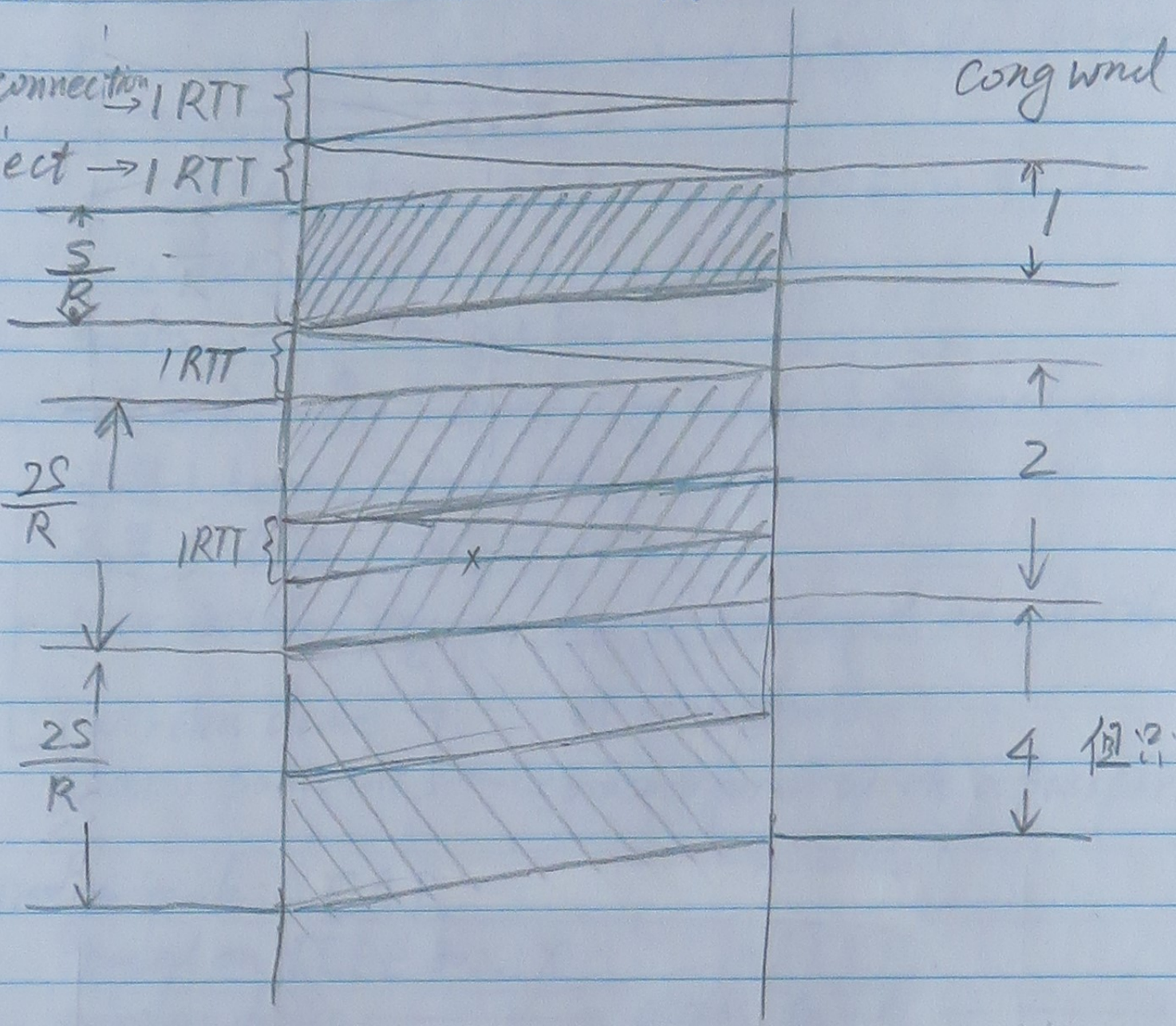
1st — 1

ack to 1st pkt ($\frac{S}{R}$) — 2nd, 3rd — 2

ack to 2nd pkt ($\frac{2S}{R}$) — 4th, 5th — 4

only when server receives ack to it 2nd pkt from receiver, the 4th pkt can start being transmitted.

(D) object's size = 5S $RTT < \frac{S}{R}$. t_3 : $3RTT + \frac{5S}{R}$.



average sending rate of a TCP connection.

$$Rate = \frac{1.2 \times MSS}{RTT \times \sqrt{e}}$$

e : loss ratio.

8.4.5 Wireless Security 802.11i

第1代技术：WEP：seriously flawed, easily breakable. ∵ 40-bit keys is too short

~~IEEE 802.11i standard~~

later：802.11i/WPA2 fixes WEP's weakness. ∵有 128-bit key

最新技术：WPA 2 即 802.11i

└ two modes ⟶ to get key

├ weaker mode：Pre-Shared Key mode.（personal mode）PSK.

├ 适用于 home，small networks.

├ 不适于企业公司，大 network,

├ user installs key as a long password on all ~~the~~ devices.

└ [wireless device / access point (AP)] — 即 passphrase / preconfigured with a passphrase

├ stronger mode：EAP

├ based on IEEE 802.1X

├ wireless device (WD) ⟷ access point (AP) ⟷ authentication server (AS)

access point forwards authentication messages among {WD and AS

├ authentication protocol：称为 EAP.

├ EAP method 包括许多不同的 authentication 方式.

├ EAP method 采用 mutual authentication

└ result of a 成功的 authentication is：a Pairwise Master Key

└ Pairwise Master Key (PM key) ┬ shared between ─ [wireless device / authentication server]

└ [access point] ⟵ send PM key ⟵ [authentication server]

└ difference：stronger mode ┬ ∵ supports a unique key per client

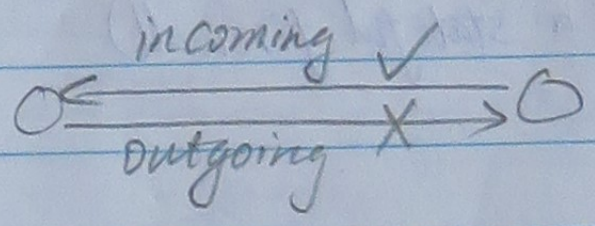
└ ∴ easier to change client

firewall on different layers

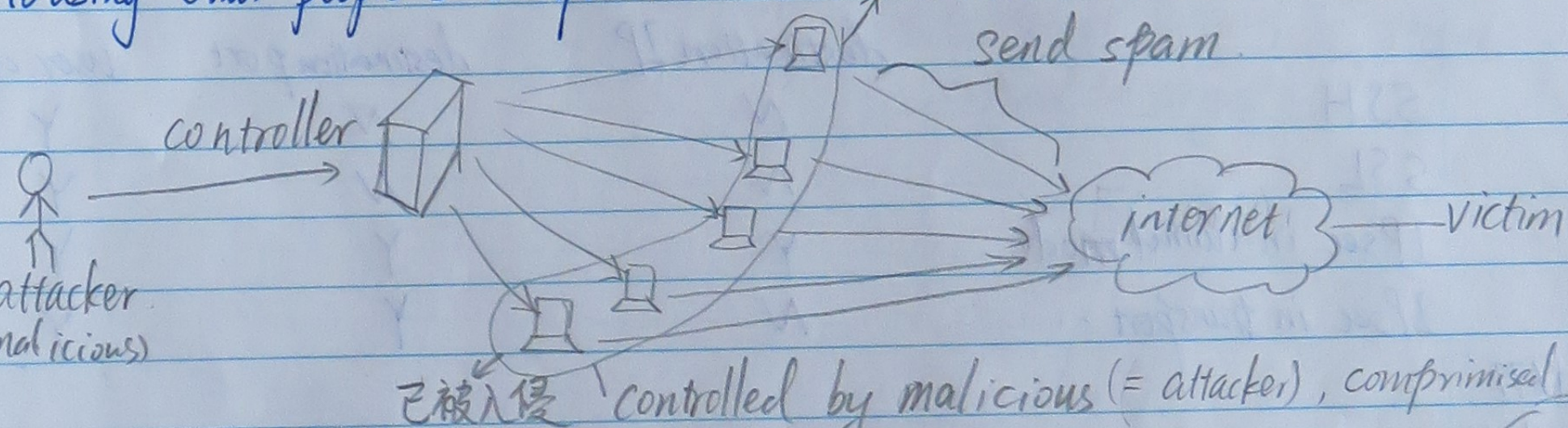
- Network-level firewall filters on IP header fields, 按 source address, dest address.
- Transport- . . . also looks at TCP fields. -- port #, flags
- Application- . . . . . contents 按 web proxy 代理 that blocks certain GET request

- stateless
  - treat each pkt in isolation.
  - no memory 关 of previous pkt
- stateful
  - keep tracks of "association" eg TCP connection, UDP flows, ICMP request
  - if pkt belongs to existing association ⇒ allow it
  - . . . not belongs to . . . . . ⇒ apply firewall rules.

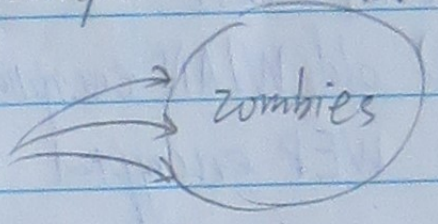
能做到/可以 can allow incoming but deny outgoing connections/associations



Abusing other people's computer : botnets (僵尸电脑) = zombies



DDoS Distributed Denial of Service Attacks



① Victim 发送正常的 request eg. 上网浏览某图片.
② 不小心 access 到 3 botnets
③ botnets 已被 malcious 告知：只需不停地向 victim 发 spam，忽视来自 victim 的 ack "message received".
④ victim 被变成 botnet.
⑤ victim 的症状 网络异常慢，无法访问 N 网站，垃圾邮件↑，服务器断线卡顿

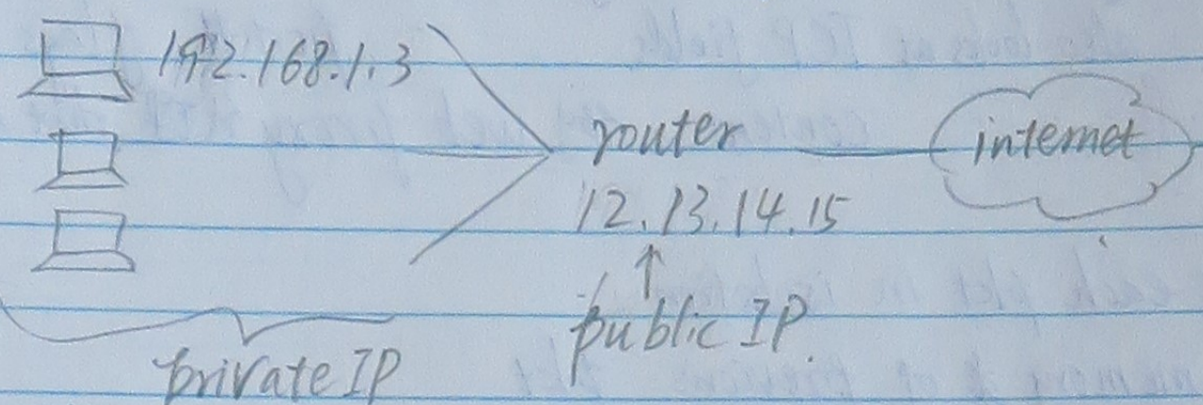
Types of Intrusion attempts

- testing what sorvices run on a computer (port scan)
- trying a lot of password to log in
- access a web server & try to exploit bugs in it to gain control
- send email containing executable code
- put code on website that exploits bugs in web browser.
- cross site script (post malicious javascript code in a forum)

IPsec may not work with NAT.

NAT vs firewalls. → translat IP & port (我老师的定义).

NAT.



for { outgoing / incoming } message

translate private IP to public IP. server only see public IP

public IP to private. . . . . .

NAT allows multiple hosts to share 1 IP address.

- outgoing do work (like a stateful firewall)
- incoming are difficult.

	what is hidden.		
	destination IP	destination port	user data
SSH	N	N	Y
SSL	N	N	Y
IPsec in tunnel mode	Y	Y	Y
IPsec in transport ...	N	Y	Y

WEP P679

(A) old WLAN encryption method WEP used keys of 40 bits. Suppose 你已 intercept 偷听到 some WEP encrypted traffic. 你 want to try all possible keys on it, hope to find the right key. your computer is fast enough to try 1 million keys/sec. 需要多少天 on average?

答: 共 2^{40} possible keys. $\frac{2^n}{2} = \frac{2^{40}}{2} = 2^{39}$

average of two extremes.

$\frac{2^{39} \text{ possible key on average.}}{1\times10^6 \text{ keys/sec}} = 550\times10^3 \text{ sec} = 6.36 \text{ days.}$

缺陷

(B) flaws are in WEP, which make it possible to find key with much less work than 以上的方式.

为了 exploit this flaw, one needs about 3×10^5 截获的 intercepted pkt. 大约需多少时间 to collect this much data on a busy WLAN: seconds, min hours or days?

答: assume WLAN transmits pkts of 500 bytes, at 11 Mbps

则 $\frac{11\times10^6 \text{ bit/sec}}{500 \text{ bytes/pkt} \times 8 \text{ bits/bytes}} = 2750 \text{ pkt/sec}$ $\qquad \frac{\text{bit/sec}}{\text{bit/pkt}} = \text{pkt/sec}$

$\frac{300000 \text{ pkt}}{2750 \text{ pkt/sec}} < 2 \text{ min.}$

(C). New WPA2 encryption: 128 bit keys. 需多久 on average 猜出 key?

$2^{128} \cdot \frac{1}{2} = 2^{127}$ keys

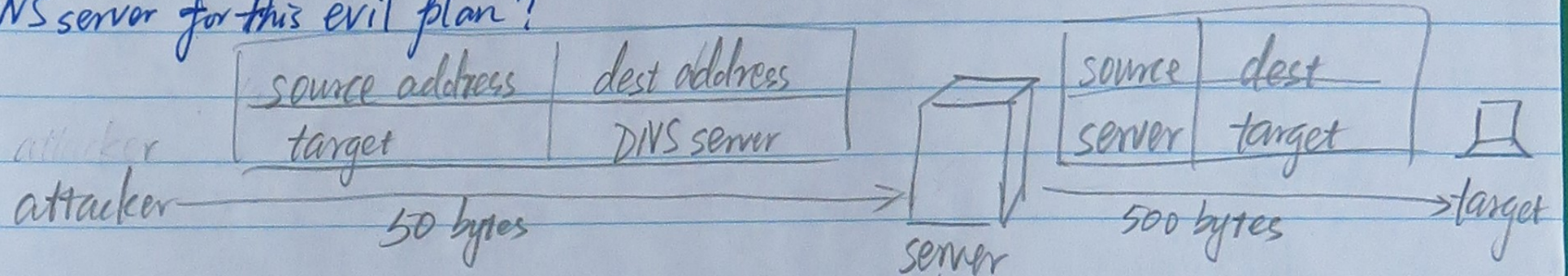
$2^{127} / 10^6 = 1.7 \times 10^{32}$ sec

keys keys/sec

Denial of Service Attack

when using DNS, a small request (eg 50 bytes) may easily 导致 a much larger response (500 bytes). ∵ DNS typically provide some extra info.

Suppose an attacker has a 20 Mbit/s internet connection at home, wants to perform a DoS attack on a site which has a 100 Mbit/s connection. how could she abuse a DNS server for this evil plan?



以为 request 真的来自于 target

∴ respond to target.