Computer Network 应用+传输层

12/28/2019

MSS: 调用方法

1. 应用层: HTTP (文件)

(1.1) HTTP 网页 ⇒ 对象 { HTML, 图片, 文件 }

用 URL 寻址 ⇒ 主机+对象路径

HTTP: 用 TCP 实现, 无状态 (不保存 client 任何信息)

(1.2) 非持续 ⇒ 一次发一个对象 (一个TCP连接) 2RTT + 传输时间

持续 ⇒ 一个TCP连接 多个对象 ⇒ 减少TCP连接打开

C: ① 请求, RTT, ② accept, Request 对象, Response 文件

非流水线: 收到一个response 发下一个request (一个文件一个RTT)

流水: 一个一个 request, 发, 回 response (一~个RTT)

(1-3) 请求报文

① 请求类型 (GET, POST), URL, 版本 (request line)

② 请求属性 + 值 (header lines) 用 line feed 隔开 (结束 <cr><lf>)

③ Body (POST 有, GET 无)

header lines ⇐

2个 <cr><lf> 结束

response: 状态 (版本+状态码+短语), 首部行, 空, Body

1.4 ① 机制: Cookies: 用于识别, 存储 HTTP 请求与响应

request → set Cookie 114514; Cookie 114514 (认证, 记录 state)

② Caches: (proxy server) (client + server)

proxy — server: 对于对象发送时间, 定期请求 + if-modified-since; 没改: 304 not modified / 修改: 不变

③ 条件 GET: 保证 proxy 保存的对象是最新的

2. 传输层: UDP: 不可靠传输

UDP + RDT: 可靠传输

2.1 RDT 1.0: 无丢包, 错误 — 发, 收

RDT 2.0 无丢包, 有错误: error detection, ACK, NAK

RDT 2.1: ACK/NAK 错 ⇒ 增加 seq: 0, 1; ACK 坏了 当成 NAK

RDT 2.2: 只用 ACK — 用上一个正确的 ACK 替代 NAK; e.g. 重复的 ACK1 替代 ACK0 — 发送0, ACK1, ACK0, 发1

RDT 3.0: 丢包: 增加 timeout 机制

分析: 等待次序: 发包坏, ACK 坏, 发包丢, 重 ACK 丢 / ACK 坏

收: 同 2.2

2.2 GBN: 窗口 — 发送: Acked, 已发 未 ACK, 未发

窗口变化 ⇒ smallest unAcked 包; ACK 发下一个, timeout: 窗口全重传

接收: 收到 seq+1: 正确 ACK; 其余 发 seq ACK 丢弃 ⇒ 只收正确包

流水线 (滑动窗口) — 窗口中的 GBN/SR

SR: 发送: 窗口: smallest unAcked; 收发

接收: 乱序包: 存储, 发 ACK

窗口: smallest unAcked 下一个

窗口大小 ≤ N/2

2.3 TCP: 端口 + seq + ACK + checksum

Sendbase: 最早未确认; 1 byte: seq+1

TCP except 当前 seq, 下个 seq = 这个 ACK

估计: $EstimatedRTT = (1-\alpha) EstimatedRTT + \alpha SampleRTT$

$DevRTT = (1-\beta) DevRTT + \beta |SampleRTT - EstimatedRTT|$ (Safety margin)

$Timeout = 4 \times DevRTT + EstimatedRTT$

seq 42 ACK 79; seq 79 ACK 43; seq 43 ACK 80 — 只 ACK 未发 已发的

重传: Sendbase: 最早的 ACK

快速重传: 3个冗余 ACK ⇒ 重传 (超时前)

流量控制: 接收方有 buffer, 剩下空间为 RcvWindow; 发送: < RcvWindow

2.4: 三次握手: SYN seq, SYN ACK, ACK (可携带 data); 四次挥手: FIN, ACK, close, FIN, ACK, closed

2.5: 拥塞控制 (网络问题): throughput 降, buffer 满丢包

TCP: slow start: 一开始1 MSS (大小), ≤ Threshold 倍增 (指数增长)

Congestion avoidance: 到 Threshold: 线性增长

3次冗余 ACK: Threshold ⇒ Cgwin/2, Cgwin ⇒ Cgwin/2 (multiplicative decrease)

timeout: Threshold ⇒ Cgwin/2, Cgwin ⇒ 1 MSS

average: 0.75 × W / RTT

TCP 公平, UDP 不公平

窗口: 一次发 < Cgwin; 吞吐量 < Cgwin / RTT

网络层 ⟶ 12/28/2019

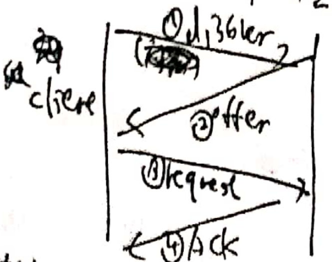
1. 区别 { 虚电路：面向连接（类TCP），预留资源，有顺序，建立、维护、拆除 —— VC表（VC号、每个链路；路由表：记录VC间转换；建立时加入新VC号，断连后清除，撤销VC号）
        数据报：面向非连接（类UDP），不预留，无状态 —— header→output link；路由表→最长前缀匹配 Forwarding table（前缀匹配：地址在某一范围的用一个链路出口）

2. Forwarding：路由器入端→出端

2.1 IP包：
- 20 bytes头（TCP 20 bytes）40 bytes（总长）
- IP版本（4 or 6）
- time to live（TTL）
- checksum（IPv6：40 byte定头，无fragmentation，无checksum，有ICMP）
- flags，fragment offset（分片fragment）
- length（长度）

2.2 IP地址：{ IPv4：32 bit，V6：128 bit } → host（路由）的IP interface；router：每个（端口一个）；子网：高位IP地址相同
- 223.1.1.0 ⇒ 分配：一开始：A、B、C类；现在：CIDR（200.13.16.0/23）
- 223.1.1.0/24 ⇒ 子网掩码 mask：前24位网络，后8位可用

2.3 IP地址分配：DHCP：动态获取



4个广播（源IP：0.0.0.0，目的地址：255.255.255.255）
① 找DHCP server
② 给个IP
③ 同意用IP
④ 确认

（前缀分配：ICANN → ISP → 机构）

2.4：内外网转换：NAT

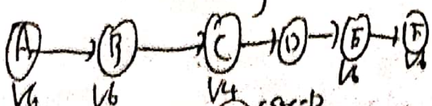
问题：外→内，内网IP不可达 traversal

NAT表：外IP + port ⟷ 内IP + port；用NAT表改包中地址（要打开传输层，改port）

- ① 静态：将客户端地址映射到NAT（e.g. 外网2500端口映射内网10.0.0.1 →2500）
- ② 动态：UPnP，IGD：动态管理外网端口与内网IP+端口的对应

2.5：一些机制：
① fragmentation：一个包分成多个 ⇒ 每个包首部包含一个IP头，标志：frag flag=0 不为最后；offset：前面包内字节数/8
（e.g.：包长为1500，内容1480，offset ⇒ 185）

② V6→V4：tunneling



V6路由A→F：B中：加包头 src=B dst=E；E中：拆包 src=A dst=F

③ ICMP：control message ⇒ type + code + description（error，ping等）（作为IP包的body）
- 应用：Traceroute
- ① time to live=1 得到IP、RTT
- ② time to live=2 得到IP、RTT ⇒ 途经路由器出，RTT变化（返回：ICMP（TTL expired），目的端口：unreachable）

3. Routing：路由器间转发 ⇒ 建立路由表（已知每个路由器的转发表）

① 全局算法：Link State（Dijkstra）（已知全部链路）
$D(w) = \min(D(v), D(w) + c(w,v))$
若 w 到 v 更新最短路，更新 v 路由表
问题：路由振荡 ⇒ efficient办法：路径非负值变化

② 分布算法：distance vector（dp）（已知相邻路由，不知全部路由表）
$d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$（邻居）
迭代性、分布式
问题：计数到无穷 ⇒ 链路长度突然变大，收敛慢
⇓
毒性逆转：若 z 经 y 到 x，则 z 告诉 y z→x = ∞，这样不会有 y 经 z 到 x

③ 结合：Hierarchical：划分AS（Autonomous System）
内网连接部分要外部连接（Gateway）
- AS内算法（不同AS可以不同）
- AS间算法
  - AS间：确定子网可达性并传播（1.0→3，1.1→3）（从1→3的子网）
  - AS内：找最近的网关，发出去（热土豆路由）（1c）

④ 实际：RIP（DV算法）
- dv算法：距离：hop：路由器到目的子网的子网数（包括目的子网本身）
- 30s 广播（advertisement）最多25子网
- 机制：毒性逆转：① 180s未收到 b 的：a认为 a→b = ∞（c(b)），并通报
- （路由器：转发表（到子网的出口，router））

链路层与局域网

appl: 应用
trans: 进程 → 进程
network: 路由 (source → dest)
link: neighbor
phys: bits

decode: dprocedure(dec·(data·trans(分析)) + dp) (信号)

12/29/2024

TSINGHUA UNIVERSITY

$d = (d_1 \cdots d_n)$
encoded data:
$Z: \; Z_{i,m} = d_i \cdot C_m$
decode: $D_i = \sum Z_{i,m} C_m = \sum Z_{im} C_m$
$= d_i$
由于各码正交: 信号叠加 完全不干扰

1. Multiple Access (MAC协议)

1.1 分
- TDMA: 分时 (每个station分时) (unused时空闲)
- FDMA: 分频 (每个station分频率) (unused空闲)
- CDMA: 分码 (编码方式) encoded signal = data $^{(1,-1)}$ × chipping sequence; decoding: encoded · chipping sequence; 多用户同时传输, 互不干扰

1.2 Random Access: 发现冲突 接收

ALOHA:
- slotted: 统一分slot, 收发 (37%) ($\frac{1}{e}$)
- unslotted: 时刻 发送 ⇒ 冲突: 冲突后下一个slot/时间有p概率重传 (18%) ($\frac{1}{2e}$)

CSMA: 发送前监听, 无人发再发 (有冲突 $\frac{d}{2}$ 传播时延) ⇒ 仍有冲突: 发送后, 但是在一段时间后发现
- CD: 冲突 Detection: 发现冲突后abort, 不发了 (有线 ethernet)
- CA: avoidance: 避免 (wifi 802.11) (无线)
  - 先发RTS包含查询 (预约信道) (广播)
  - 收到后 发送 CTS sender发送, 其余 defer
  - 空闲: 先 DIFS 再 发 ⇒ 完全收到 SIFS后 ⇒ ACK

1.3 taking turns
- poling: master-slave: master让slave轮流发
- token passing: token 轮流在各点中传, 有token的发

switch: 无路由 (不改MAC) (透明)
交换: 自学习 - 记MAC

2. 访问地址: MAC地址 (48位, 写死在硬件)

MAC与IP对应: ARP ⇒ IP node 均有 ARP表 (MAC与IP对应, 有TTL: 20min)

A 广播 ARP query (mac: 全 8个FF) 带有B的IP
B 收到后回复 ARP (发到A的mac地址, 带B的IP)
A 存储 {B: IP, B: MAC, Timeout}

跨LAN: A先找网关路由的IP, 存 ARP ⇒ A发给R转发; R查ARP找B IP, 存 R再转发

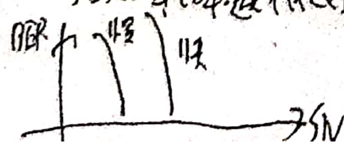
3. 具体协议

	以太网 (一般, 有线 介绍)	WIFI 802.11
拓扑	star topology (中央有switch)	也中央有switch ⇒ 由多个BSS (Basic Service Set) (host与AP相连接定) (每个AP通过DHCP分配IP) 每个BSS一个AP (DS, 基站), 一大块区域覆盖
包结构	preamble: 用于同步时钟; src, dst 的MAC地址, type; IP包, crc (校验)	control, sequence (用于RDT); IP包, crc 校验; 4个address: (src, dst, AP, ad-hoc模式) (MAC)
multiple access	CSMA/CD ⇒ exponential backoff: m次冲突后, 等 K·512 bit times, 其中 K从 0 → 2^m 随机取	DIFS, Data; CSMA/CA (现代网络: e.g. A→B可达, A→C可达, A与C不可达) (隐藏终端问题)
入网, 获取IP	ARP 得 IP	入网: 被动: AP发beacon frame (host 从收到的中选); 主动: host广播, 收到的AP都回复, host选择, 确认
		地址: AP-router间只有收发2个; host间: AP, src, dst 3个 (Ack 可) Hidden terminal (AB间相互不见) decrease signal length

bit time: 10Mbps 下 为 0.1 us

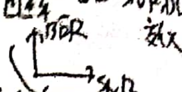
移动: IP不变, switch: 自学习, 记住传输的H1

SNR: 信噪比, 越高越好
BER: 出错率, 越低越好



距离变远: SNR下降, BER上升; BER大: 选慢点

特性2
① Rate Adaptation: 动态切换发送速率 以确定 SNR BER 较大



② power management: node告诉AP要sleep到下一个beacon frame, AP存储; beacon frame: AP给node发所有包

网络安全　　12/30/2019

攻击: sniffing (窃听), IP spoofing (伪造IP), record and playback (重放)

截获, 伪造报文, 修改, 重放

1. 主要需求: { confidentiality: 加密, 仅发送方和接收方明白; authentication: 确认身份 }
integrity: 未修改
access, availability: 服务可用的

2.1 对称加密: 一个密钥加解密 K_s

流 (stream): 一次加密 1bit
$c(i)=k_s(i)\oplus m(i)$ 密文
$m(i)=k_s(i)\oplus c(i)$ 明文
密钥

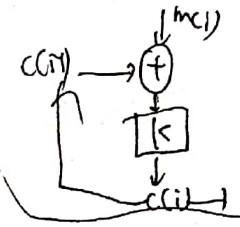
分组块 (block cipher): 一次加密一块 (k-bit 定长) (常用)
块替换 一对一映射
(2^k)! 种映射, 难以暴力
但 2^k 太大, 表太大
⇐ K 不太大 + permute

prototype:
明文 64bit 切分为 8个 8bit
用 S_1—S_8 分别加密
permute
循环加密

DES: 加密 64bit: 16轮, 每轮 permute + 加密, 每一轮密钥不同
AES: 128bit

加密大量信息: ⇒ { 拆分成 64-bit; 随机数 r 公开 }
$c(i)=k_s(m(i)\oplus r(i))$, $m(i)=k_s(c(i))\oplus r(i)$ 随机编码

CBC (block chaining): 只需一个初始化 c(0)
$c(i)=k_s(m(i)\oplus c(i-1))$
$m(i)=k_s(c(i))\oplus c(i-1)$



{ 重复 ciphertext 也不同; 每轮加密不同 c(i) }

2.2 非对称加密: 公、私钥 K^+ K^-

比对称慢的多

公钥: 加密用; 私钥: 解密

RSA: p, q 大质数
$n=pq$, $z=(p-1)(q-1)$
e 与 z 互质, $ed\%z=1$
公钥 (n, e)　私钥 (n, d)

$m^e \% n$
$c^d \% n$

性质: $K^+(K^-(S)) = K^-(K^+(S)) = S$

①: 加密: 公钥加密, 私钥解密
② 数字签名: 私钥加密, 公钥解密验证

2.3: 摘要 message digest (哈希) (单向的)

integrity

message authentication code (MAC): $H(m+s)$ (m, 密钥)

2.4 数字签名

authentication

私钥加密, 公钥解密
signed message digest $K_B^-(H(m))$

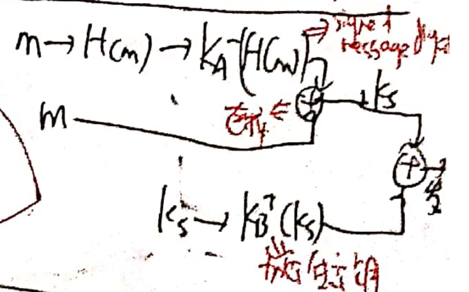
⇒ 在 CA 进行公钥注册认证
$K_{CA}^-(K_B^+)$: CA 认证的公钥
K_{CA}^+(认证) $=K_B^+$ (正确的)

防篡改, 防抵赖
MAC = f(msg, s, R)

2.5: 应用层加密

应用: { 对称密钥 K_s 加密报文; 用公钥 K_B^+ 加密 K_s; H(m) 做数据摘要; 用私钥 K_A^- 做认证 }

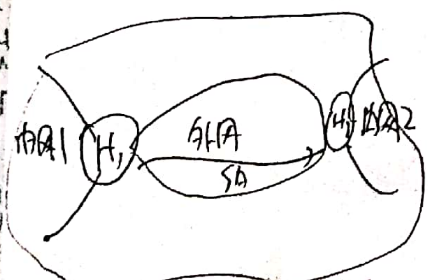
安全, 性能, 认证, 算力 (对称 << 非对称)



3. IPSec

IP层加密; VPN
AH, ESP (加密、认证、完整)
transport, tunnel

① SA: 单向 (存在 SAD)
建立逻辑连接 存 (源、目的 IP, 接口, SPI; 加密、认证算法, 密钥)
SA 的 identifier, 32bit



防重放攻击: 认证, seq

② IPsec包:

新IP头	ESP头	旧IP包	ESP尾	ESP auth

地址: 网关地址 (地址 网关 IPsec)
ESP头: sequence (防重放), SPI → SA认证
加密: ESP尾: padding, length, 下一个头 — 旧IP包的协议类型
padding: 用于块加密需要
签名: ESP头 + IP包 (enchilada)

① 发出步骤:
① 旧IP包 + ESP尾 + padding → 加密
② 加ESP头 { seq 防重放; SPI: SA认证 }
③ ESP头 + IP包 (enchilada) ⇒ MAC 加密

收:
①: IP: 是否 IPsec
② 验证 SPI → SA
③: MAC 验证
④ 序列验证
⑤ 解密, 去 padding

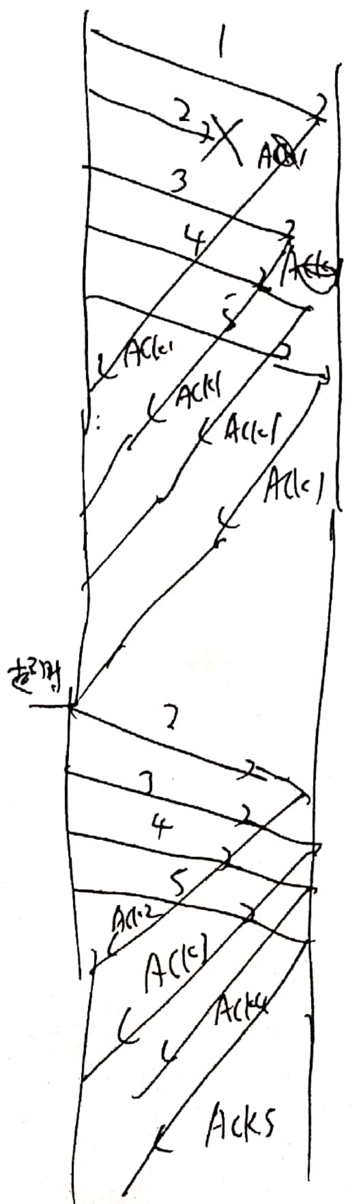
④ 建立: SPD 策略库
IKE: 公钥系统
PSK: 共享密钥
PKI: 认证机构公钥

清華大學
TSINGHUA UNIVERSITY

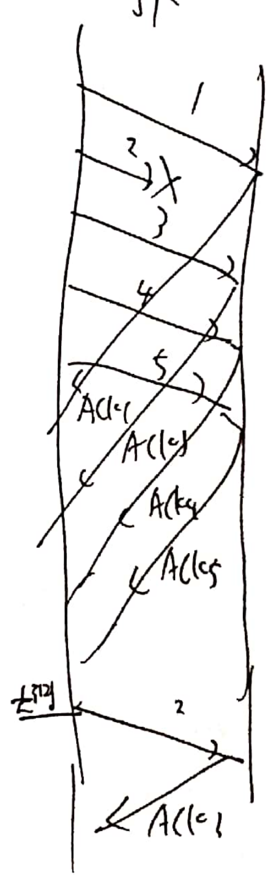
12/29/2019

传输 P37：发送方向接收方发包 1,2,3,4,5. 包2发送时丢了. 包(ACK)回来时超时.

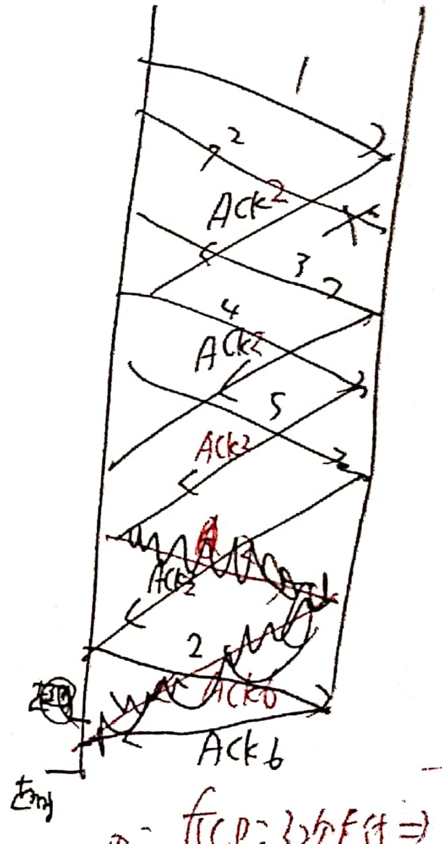
GBN



SR



TCP



①：TCP：3次冗余 ⇒ 5的ACK2 回来：自动重传2，未到超时

②：TCP 返回的 ACK=预期 (+1)

P4：

MSS=536（应用层包大小 ∴ +66）

trans the work of link：66 bytes

4G文件，155Mbps 所的传输时间

共 $\frac{4\times1024^3}{536}$ 个 2 8012999个包

$\frac{4823825398 \text{ bytes}}{\frac{155}{8}\times10^6 \text{ bytes per second}} = 248.97$

P46: 10Mbps链路

RTT: 0.15s

一个报文: 1500 bytes

①: 最大窗口　　1s: $\frac{10\times10^6 \text{ bit/s}}{8}$ bytes ⇒ 1个RTT: $\frac{10\times10^6\times1.5\times10^{-1}}{8}$ bytes ⇒ $/1.5\times10^3$ bytes ⇒ 125

②: 平均窗口,吞吐量 ⇒ 平均 $125\times0.75=94$

吞吐量: $\frac{94\times1.5\times10^3\times8}{1.5\times10^{-1}} = 7.52$ Mbps

③ 丢包一次后恢复时间 ⇒

最小: $\frac{125}{2}=62$ (1/2)

最大: 125

$63\times0.15s=9.45s$

网络层

R15　经了个router, 有 8 接口, 3 转发表

P10, 11, 17: 最长前缀匹配 ⇒ 表 ⇒ 子网

0000 0000 – 0011 1111	⇒ 0	
0100 0000 – 0101 1111	⇒ 1	
0110 0000 – 1011 1111	⇒ 2	
1100 0000 – 1111 1111	⇒ 3	

⇒ 前缀:

00	⇒ 0
010	⇒ 1
011	⇒ 2
10	⇒ 2
11	⇒ 3

⇒ 子网:

0.0/2	⇒ 0
4.0/3	⇒ 1
6.0/3, 8.0/2	⇒ 2
12.0/2	⇒ 3

范围 前缀 ⇒ 子网: 直接(前缀上一)

范围 前缀 ⇒ 范围 (由长到短)

范围 ⇒ 前缀, 找到一个合适 拆分

P19: 分片

标识 422

链路 700 bytes

数据报 2400 bytes

①: [illegible] 一个包实际 680

总共实际: 2380

2380/680 ⇒ 4

4个包

大小: 680 680 680 340

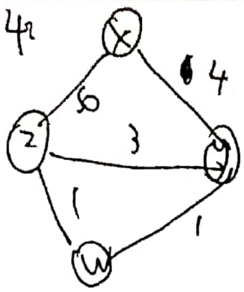
⇓ ⇓ ⇓ ⇓

700 700 700 360

标识	长度	flag	偏移
422	700	1	0
422	700	1	85
422	700	1	170
422	360	0	255

12/30/2019

P34



(有毒性逆转)

①：稳定时：w的距离 (∞, 1, 1, 0)

z的距离：(6, 2, 0, 1)

解：表如下：

	x	y	z	w
x	0	4	6	5
y	4	0	2	1
z	6	2	0	1
w	5	1	1	0

w→x: w→y→x

w→y: w→y

w→z: w→z

z→y: z→w→y

z→x: z→w→y→x

z→w: z→w

w→x 直接过 y: ∞

z→x 过 y 仍为直接过 6

② x→y 变化：

毒性逆转有效？

更新 $d_y(x)$

$=\min(d_x(x)+c(y,x),\ d_z(x)+c(y,z),\ d_w(x)+c(w,y))$

$=9$

实际：52 ✗

仅某环路 毒性逆转无用

失效

收敛：①：$d_y(x)=9$ $d_z(x)=6$ $d_w(x)=5$

②：$d_z(x)=\min(d_x(x)+c(z,x),\ d_w(x)+c(z,w),\ d_y(x)+c(z,y))=6$

$d_w(x)=10$

③

11/1/2020

链路 P18:

传播时延 325 bit time
在t=0时刻A发，B开始传输
(最坏) 问A能否在t时刻冲突前传完

解: B检测到A冲突: 325
B在324传输
A检测到B: 又325 bit
649 bit
A总共: 512+64 = 576 ⇒ 一帧
576 < 649
能传完

RF: 802.11 发数据前必须先发RTS和CTS (X)
(重传!)

安全: CBC:

明	密
000	110
001	111
010	101
011	100
100	011
101	010
110	000
111	001

无CBC
010 010 010 → 101 101 101
有CBC，初始 001:
$c(1) = K_s(c(0) \oplus m(1)) = 100$
$c(2) = K_s(c(1) + m(2)) = 000$
$c(3) = K_s(c(2) + m(3)) = 101$
⇒ 100000101

只有 K_s

解: 先 K_s 再 ⊕