

1. (1) 平均队列长度、缺乏缓冲区造成的丢包率、链路利用率
- (2) TCP, UDP
- (3) 端口号
- (4) 应用层
- (5) 字节
- (6) 传输层
- (7) 500
- (8) 1KB
- (9) 端口号
- (10) 接收方窗口(rwnd) 拥塞窗口(cwnd)

2.

- | | | | | |
|-------|-------|-------|-------|--------|
| (1) A | (2) B | (3) C | (4) C | (5) B |
| (6) C | (7) B | (8) C | (9) D | (10) C |

3.

(3) 拥塞是通信子网中信息量太多, 导致性能大为下降的现象

原因: ① 多个输入对应一个输出

② 慢速处理器

③ 低带宽线路

(4) 关系: 都是为了保护网络数据传输的高效性和可靠性, 防止出现问题。流量控制是拥塞控制的基础, 拥塞控制从更宏观的网络层面进行的控制

不同点:

拥塞控制：需要确保通信子网能够承载用户提交的通信量，是一个全局性问题，涉及主机、路由器等很多因素

流量控制：与点到点的通信量有关，主要解决快速发送方与慢速接收方的问题，是局部问题，一般都是基于反馈进行控制的。

(5) 不能，

因为网络拥塞是由于网络资源有限，而数据流量超过网络负荷导致的；且网络流量动态变化，可能出现突发情况等不可完全预测的情况，所以算法只能缓解，无法彻底阻止其产生。

(7) 当路由器监控新包的输出路线是否处于警戒状态，如果是向源主机发送抑制包，指出发生拥塞的目的地址，并将原包打上标记，正常转发，源主机收到抑制包后按一定比例减少发送，并在固定时间间隔内忽略抑制包，开始监听是否拥塞，如果是，固定时间减少负载，如果没有收到抑制包，则增加负载。

(8) 不能

IP协议负责网络层的分组转发，没有对数据的完整性、顺序性提供保障，也缺乏端口寻址能力。但是UDP提供端口号，能区分不同进程，对数据进行一定的封装，更适用于应用层的需求。

(10) 保证通信的可靠性，确保双方收发能力正常，序列号同步可靠。

第一次握手：客户端向服务器发送SYN报文，请求建立连接

第二次握手：服务器收到客户端的SYN报文后，发送SYN+ACK确认

报文响应，既确认请求，又向客户端发起同步请求

第三次握手：客户端收到服务器的SYN+ACK报文，发送ACK报文确认。

第一次挥手：客户端发送FIN报文，请求断开连接，不再发数据

第二次挥手：服务器收到FIN报文，发送ACK报文确认

第三次挥手：服务器发送完数据，发送FIN报文，请求断开连接

第四次挥手：客户端收到FIN报文，发送ACK报文确认

因为TCP连接是全双工，关闭需要关闭两个方向的数据流。

在第二、三次挥手为什么不合并，因为服务器可能需要发送数据，发送完再发送FIN报文。

(12) ① 数据处理更加灵活，可根据网络情况灵活地将字节分割为大小不同的报文段

② 流量控制更精细，通过序号精确进行流量控制，接收方能够明确告知发送方自己期望接收的下一个字节序号

③ 对应用层透明，应用层向TCP发送数据，无需关心数据被如何分段。

(13) 超时重传是判定拥塞的一个重要标准

超时重传判定拥塞后的处理：发送方将cwnd置为1，进入慢启动，缓慢增加cwnd。

重复确认判定拥塞后的处理：发送方会执行快速重传，立即重传丢失的报文段，同时执行快速恢复，将cwnd设为当前一半，线性增加cwnd，逐渐恢复。

4.

(1) 令牌桶容量 + 令牌输入速率 $\times t$ = 数据最大输出速率 $\times t$

$$t = \frac{\text{令牌桶容量}}{\text{数据最大输出速率} - \text{令牌输入速率}} = \frac{250 \times 1024}{23 \times 1024 \times 1024}$$

$$t \approx 10.6 \text{ ms}$$

(2)

① 30个字节

② 100

③ 80个字节

④ 确认号是 70

(3)

第一个RTT, 发送窗口为 $2 \times 1 \text{ KB} = 2 \text{ KB}$

第二个RTT, 发送窗口为 $2 \times 2 \text{ KB} = 4 \text{ KB}$

第三个RTT, 发送窗口为 $2 \times 4 \text{ KB} = 8 \text{ KB}$

⋮

第五个RTT, 发送窗口为 $2^5 \times 1 \text{ KB} = 32 \text{ KB}$

$5 \times \text{RTT} = 25 \text{ ms}$, 经过 25 ms 到达 32 KB 。