1. 原码、反码、补码

正数全一样，负数反码：符号位不变，其余取反； 负数补码：反码+1

引入反码原因：加法代替除法，计算机加减自带取余操作。 2-1 =2+（-1）= 2+126 mod 127 = 1 mod 127

引入补码原因，消除正负0，且只是相当于增加了模的值。

1. 进程的创建：

（1）系统初始化（2）系统调用（3）用户调用

linux下 ： pf – aux 查看所有进程；

1. 虚拟内存

**核心思想：通过讲程序**

虚拟内存提出的原因：

（1）（2）（3）

1. TCP拥塞控制

引入原因：防止出现网络拥塞，大量丢弃发送的包

四个阶段：

首先是发送方维护一个拥塞窗口，窗口内进行连续发送

1. 慢开始。指数增长。
2. 拥塞避免。当窗口值大于阈值后，线性增长。出现丢包时，则讲阈值改为现有值的一半，窗口值从一开始

但是丢包不一定代表网络拥塞，优化：

1. 快重传。发送方和接受方都按序进行。接收方如果收到乱序包，则重复发送最后一个有序包。发送方如果多次收到某个包的确认，则认为发生丢失，进行重传。接收方收到后只用回复最后一个有序包确认
2. 快恢复，此时阈值修改为一般，且窗口也从一半再次开始线性增长。再优化，窗口从一半+3开始。ack冗余重复确认包为3，已经在接收端了，所以适当增大。

**TCP协议**

1. 定义：
2. TCP与UDP区别
3. TCP三次握手过程及原因(原因：1.防止历史连接2.防止资源浪费3.确定序号)
4. MSS和MTU解释，TCP层和IP层分片
5. SYN攻击
6. 四次挥手过程及原因
7. time\_wait存在原因如果过长的危害

TCP机制 小林coding P158-190

1. 重传机制
2. 滑动窗口机制

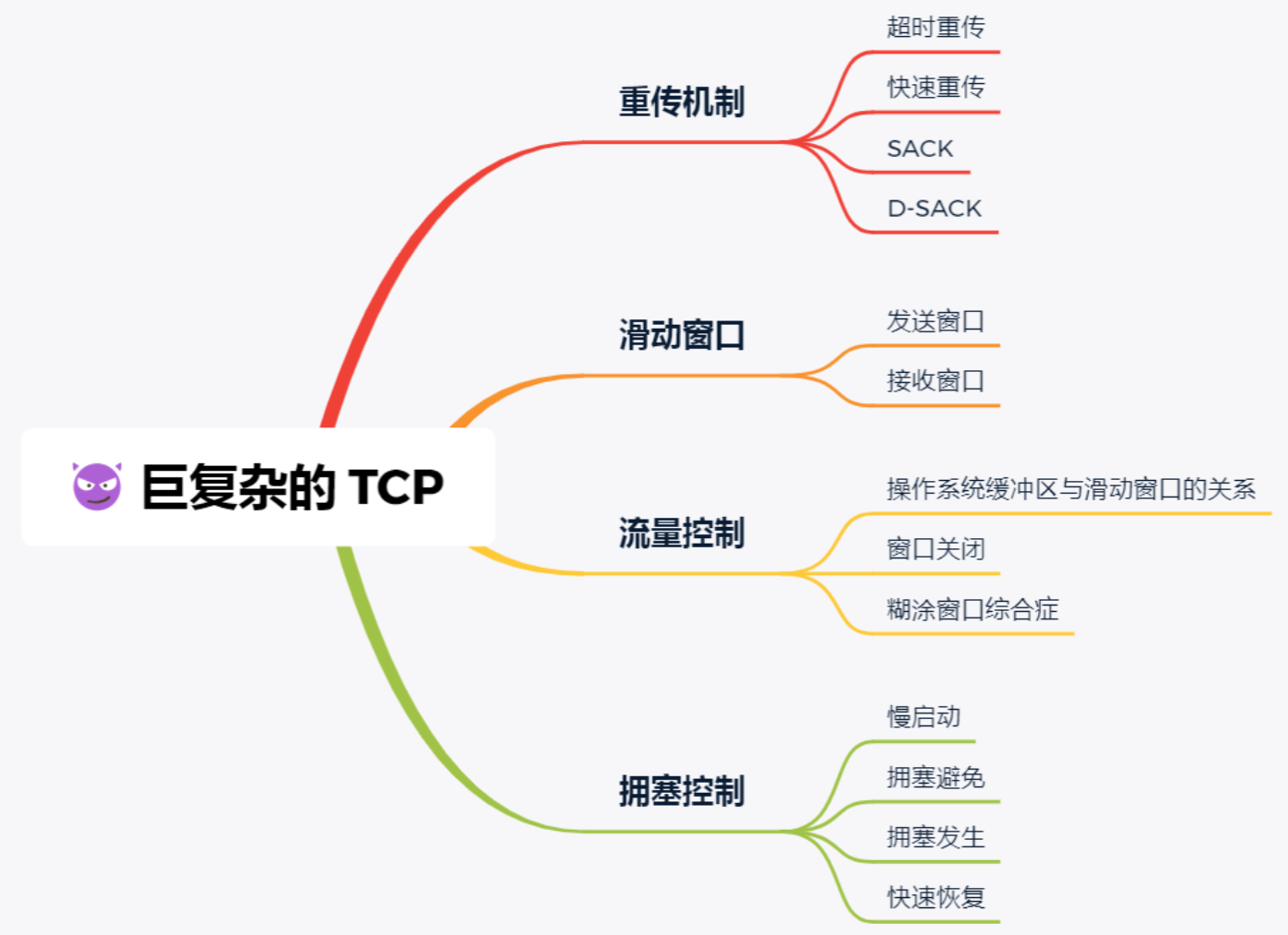
三种窗口：

引入原因：

机制：

TCP潜在死锁：

糊涂窗口综合症:



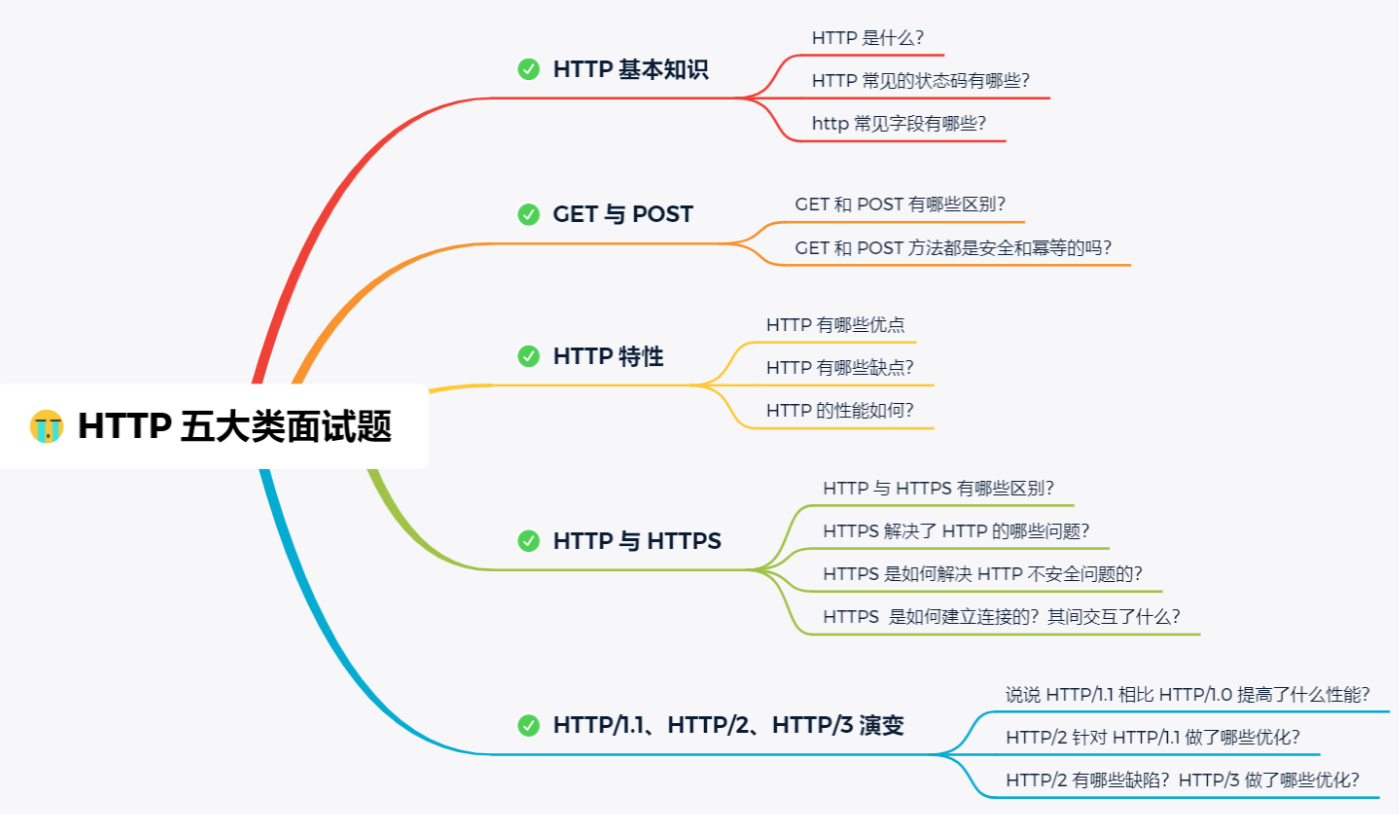
（1）重传机制。超时+快速(三次)。SACK(响应接受发缓存情况)。D-SACK（1.接受方丢包=超时+SACK<ACK;说明收到了但是没有回应） 2.网络延迟=快传+SACK<ACK；说明先没有收到，后面收到了）

（2）滑动窗口。发送方（1.数据分段2.两个指针+一个变量）。接收方（1.数据分段。2.一个指针+一个变量）

（3）流量控制。（1.探测 2.糊涂窗口）

（4）拥塞控制。1.慢开始（刚开始小，比直接慢得多）

1. HTTP



补充：

HTTP报文组成（行+头+正文）、报头常用字段（Host、Connection、Content-Length、 Transfer-Encoding、Content-encoding、 content-type(Accept)）

请求行= 方法+url+协议 响应行=协议+状态码+原因

设置connecting为keep-alive 建立长连接，使用content-length的响应包长度告诉浏览器响应包结束情况，遇到大文件时通过设置transfer-encoding 为chunk来分段响应。

无状态：验证问题，通过cookie技术解决。服务端返回一个cookie，客户端第二次访问带cookie由服务器验证并返回新的cookie.理解为通行证。 或者用Session解决，服务端维护。为每个客户端建立一个Session\_id保存在cookie中。

https增加的三个主要功能：加密，校验、证书

**#**http1.1优点：长、管 ；缺点：未压缩冗余、队头阻塞、无优先级请求

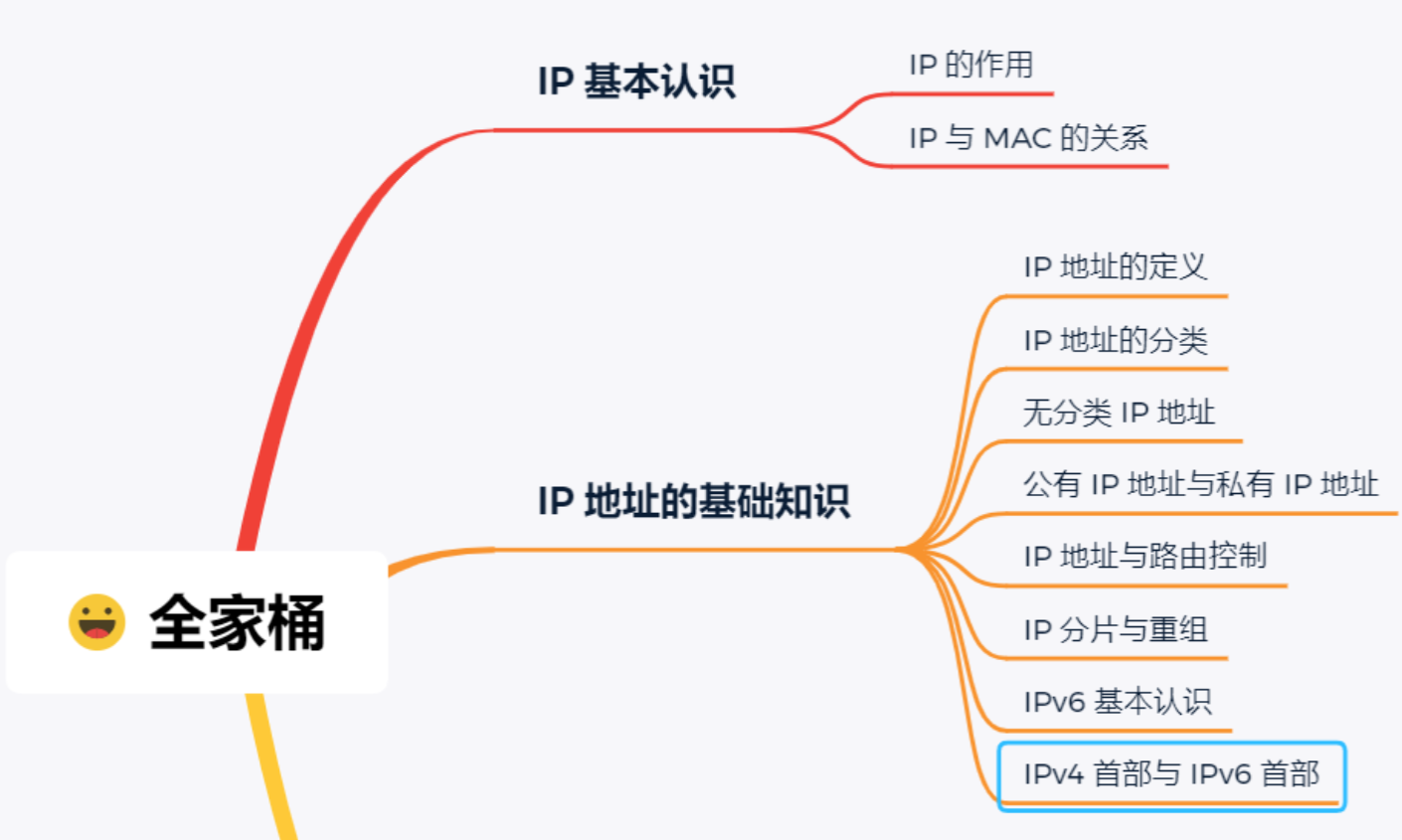
**#**http2.0优点：头部压缩、二进制、数据流、多路复用、服务器主动推送；缺点多http公用tcp，丢包时全等待

#http3.0优点：基于UDP，使用QUIC协议实现伪TCP效果。合并tcp三次和TSL三次握手。

GET和POST的区别：

是针对应用层的协议，对应了读服务器资源和写服务器资源，都是用tcp包发送请求。主要区别在于：（1）请求信息存放位置：GET存于URL中，?号后的内容，用&隔开。POST存在于请求体正文中。（2）GET有长度限制，POST则没有（浏览器应用限制的）（3）POST更安全一点，浏览器会对GET请求内容缓存,服务器会将URL存入日志。POST则不会（4）大多数时候GET更快一点，POST会先传请求头再正式传输。

GET适用场景：希望被缓存，被搜索引擎收录，可以存入书签或分享给他人。



1. 直接广播和本地广播，全1用于广播，全0代指对应网段
2. 多播？
3. 五类地址判断方法：



1. **路由协议**

**黑洞路由 ：**NULL0接口，用于丢弃ip包。用处：过滤，防环

**EGP:**域间路由协议，主要有BGP

**IGP:**域内路由协议包括RIP\RIP2\OSPF等。

**RIP协议：**路由表维护ip/距离/方向 这三个主要参数。路由器不断将自身的路由表发送给相邻路由器，并从相邻路由器接受路由表进行更新。优点：简单。缺点：收敛慢，容易造成环路。（比如A/B互相捧臭脚）。

采用**距离向量算法**：具体做法：路由器收到相邻路由器X的RIP报文后，将下一跳改为X，然后所有距离+1。然后更新路由表。如果路由表中没有该网络M，直接加入该信息。如果有，且旧的下一条为X,则直接更新距离。如果不是X，则选择更短的加入。

**OSPF协议：**使用泛洪的方法，使所有路由器维护相同的链路状态（该路由器相邻的节点，以及到达节点的“开销”），最终建立全局的网络拓扑图。并以自己为根计算最短树。然后只在链路状态更新时再次改变链路状态。优点：收敛快。缺点：消耗内存和cpu。

且OSFP协议可以自定义代价，还可以做负载均衡。

**BGP 协议：**域间路由协议，选择最想要的路径。BGP发言人之间通过建立TCP连接来交换双方域内的信息。同时发言人除了允许本身的BGP协议外，还需要运行域内路由协议来掌握域内路由信息。

1. **DHCP协议**

DHCP Discover: from 0 to 1。使用udp广播

DHCP Offer: from sever\_ip to 1。使用udp广播，发送ip,网关，掩码，以及ip租用期。

DHCP Request:选择一个回复配置参数

DHCP ACK：响应确认

**DHCP 中继：**DHCP管理不同网段。每个网段设立一个DHCP中继，客户端通过广播向中继请求，中继发送单波向DHCP服务器请求。

1. **NAT协议**

私有IP和公有IP，主机和端口绑定使用同一个公有ip.

1. **ICMP协议**

**符号执行**

分为过程内分析和全局分析（需要考虑函数入口和环境）

目的：将程序输入用符号表示，则输出得到符号表达式。不同条件产生不同的分支，使用程序流程图来进行表示，通过符号执行可以覆盖整个程序的各个分支。（比如一个程序含有三个分支，我们输入三组数据，就可以到达这三个分支，符号执行的目的就是为了找到这三组输入情况。）

方式：全局维护两个变量：符号映射d和路径条件记录PC。两个变量在符号执行过程中不断改变。其中d在遇到新的外部输入以及赋值语句是会进行状态改变。而PC则是在遇到条件语句时进行状态更新，对于遇到的条件分为两个分支，产生新的符号示例，拥有不同的路径约束。遇到exit或程序错误结束符号执行。

举例

test(x int, y int){

z = twice(y) //如果twice函数未知，则无法解决。

if z>x{

}

}

动态符号执行：

目的：解决约束器无法高效求解的问题

方法：（1）动态符号执行维护同时维护一个实际状态和符号状态。针对程序的执行流程图，使用了一种深度优先遍历的策略。将实际状态和符号状态结合，对实际状态的执行路径上的约束条件进行取反，得到新的实际状态，不断的递归约束，直到遍历完所有可达路径。

举例：上述函数，x,y 同时随机生成一个{x=4,y=3}。在遇到x==2y条件时，实际状态走到else路径，然后符号执行会新加一个条件约束{x!=2y}。然后为了执行实际状态对应的另一个分支，会对这个条件取反，{x==2y}得到一个新的实际状态{x=2,y=1},然后继续实际执行和符号执行共同进行，遇到新的条件继续求反以执行新的路径。直到路径探索完或超出限制。

优点：简化了约束求解器，对于非线性条件和外部函数调用可以有效解决。比如一个条件时x==y^2mod50。通过实际执行可达路径为y=4,x=3；则用实际值代替不可求解的表达式，则条件变为了x==16。就执行了一条路径。但是另一条路径是无法到达的。

1 2 3 4 5

TOP K问题

linux分段：

.bss 未初始化全局变量； .data 已初始化全局变量；代码端；堆（动态分配）；栈（局部变量）

redis常见数据结构：

1. 字符串：缓存，作为持久化数据库（如mysql）的缓存。计数，共享session等
2. 哈希表：更加直观的缓存。可以存更多的信息，如用户信息、视频信息等
3. 列表：类似于链表或者数组，用于时间轴的实现。比如微博中，通过lpush将新的微博加入以此在列表中展示。
4. 集合：无序不重复的。可以用于给用户打标签，用户喜好推荐。用于存储点赞点踩，收藏等信息。
5. 有序集合zset：有序的，可以用于实现实时的热搜榜，排行榜等。

redis持久化策略：

1. RDB数据快照模式（2）AOF 追加模式

redis删除策略：

1. 定时删除（为每个key加定时器）（2）惰性删除（被动删除）（3）定期删除（隔段时间进行扫描）

redis具体采用策略：惰性删除+改良定期删除（及每次随机抽查，删除一定数目的过期key）

redis缓存穿透、击穿、雪崩：

1. 穿透：缓存和持久层都不存在，不会返回缓存到redis，导致每次访问都会到mysql。

解决方法：布隆过滤器（将所有可能的key（比如正值）进行hash存入布隆过滤器，每次请求前先判断）；或者返回一个有时间限制的空值缓存。

1. 击穿：对一个存在但过期的key的高并发

解决方法：设置key，操作dp前上锁，执行成果返回缓存后在进行其他的操作。

1. 雪崩：redis挂了

解决方法：使用高可用redis，如分布式集群；设置过期时间，防止集体过期；本地缓存。

redis的集群模式：

1. 主从模式：1主n从。主负责写，从负责备份和读（如果可以写则不更新到其他节点。）
2. 哨兵模式。基于主从模式，哨兵进程监听，主节点挂了自动竞选主节点，并通知。

redis一致性问题：

主要针对双写问题。

1. cache-aside.旁路缓存模式
2. read/write-through 读写穿透模式
3. write-behind 异步缓存操作
4. 先更新数据库再更新缓存：线程AB导致脏数据问题。
5. 先删除缓存再更新数据库：A删除更新数据库。同时B读取缓存为空，在A更新数据库前拿到脏数据再更新缓存。

（#1）先更新数据库，再更新缓存:A写mysql，B写mysql+redis，A再写redis导致不一致

（#2）先删redis，再写数据库：A删redis，B读redis+mysql并更新redis，A再写mysql

（#3）先写mysql，后删redis：A写mysql，B读redis命中，A再删redis，短暂的不一致。

MySQL innodb索引：

1. B+树，优势：（1），结点只存索引，高度更低（2）空间更小（3）更有利于遍历（4）查找效率更加平衡稳定。
2. Innodb主键索引采用聚簇索引，其他采用非聚簇。优点：访问快，利于排序和范围查询。缺点：插入速度依赖于插入顺序。更新主键代价大。

MySql高可用：

主从、MMM(互为主从)

乐观锁：读写不加锁，操作完再判断

悲观锁：所有操作都加锁。

Innodb关键特性：

一致性hash：

（）1比如将不同的文件存入redis，如果不hash需要遍历所有的redis服务器。对key进行hash，可以得到所在redis服务器的位置。（2）但是如果redis服务器发送变动，就会造成hash值变化，所以引入一致性hash。通过对服务器本身也做hash，然后将其映射到0~2^32-1的环状空间中。key使用相同的hash映射到该环状空间。按照顺时针方式，遇到的第一个节点就是key所属节点。好处：服务器改动时只变更小部分数据。（3）但是可能会出现数据倾斜的问题，所以引入虚拟节点。每个服务器做多个hash作为虚拟节点，然后进行映射。可以做到少量服务器也可以数据较为均衡。

**Oyente相关**

1. 智能合约。智能合约通过字节码的形式部署于区块链上，用户通过交易的形式与智能合约进行交互。具体来讲，如果一笔交易被矿工打包进入区块链，则网络上所有参与者都需要以交易中函数的有效参数以及当前区块链的状态去执行合约，并得到一个相同的结果。
2. 智能合约通过交易来调用和执行（相当于是各类函数调用）。智能合约执行时，用操作栈、内存m, storage S来保存相关信息。其中S会最终保留。可以用一个五元组{g,pc ,m,s,st}来表示当前虚拟机的状态。那EVM中的操作码就可以表示为：一个函数，作用于虚拟机状态d,使其状态改变为d’。

符号执行 5 将程序变量的值表示为输入符号值的符号表达式。每个符号路径都有一个路径条件，它是符号输入的公式，该公式是通过累积这些输入必须满足的约束来构建的，以便执行遵循该路径。如果路径条件不满足，则该路径是不可行的。否则，路径是可行的。

Oyente主要包括了三部分：CFG Builder, Explorer , Core Analysis

其中Oyente通过调用官方编译器solc将bytecode反汇编为汇编指令，然后模拟以太坊虚拟机的执行过程，对操作码进行执行。其中通过CFG Builder建立控制流程图，将汇编指令分块，并进行一个函数的映射。然后Explorer进行符号执行。具体的执行过程为：通过CFG流程图进行执行，维护一个状态，不断的执行指令，同时修改这个状态。在遇到JumpI条件条件指令时，相当于一般程序中的if条件语句，修改当前条件约束，然后循环执行到结束；得到各路径的约束条件后，在使用Core Analysis模块使用z3求解器进行约束求解，判断各类bug的条件是否可达。

**项目一DeFi总结：**

1. 项目意义：针对以太坊上的安全漏洞，特别是以太坊上的DAPP即区块链应用进行安全检测。DAPP涉及到多个的智能合约，现有的很多攻击是很多合约共同作用的结果，现有方法无法满足。我们提出了新的方法来对DAPP进行检测。
2. 项目流程：针对DAPP中应用最广泛的DeFI项目进行检测，针对多种攻击提出了不同检测方法。（举例：重入攻击、套利行为、前端钓鱼攻击等）。DAPP中使用到的合约会记录一些用户的变量，这些变量理论上应该保持一致。我们通过插装代码，获取到不同合约的关键变量。具体做法：首先用符号执行预先得到所有变量在合约中的位置，然后通过EVM虚拟机的操作码序列定位变量，根据Sstore操作获取改变量进行比较。
3. 项目代码框架：
   1. 插装代码。

####geth工作流程：通过创建一个EVM虚拟机来执行函数调用。具体过程为：1.每笔交易创建一个EVM的环境，配置相关参数 2.根据交易地址，从stableDB数据库获取到所用合约的代码 3.调用call函数作为执行入口，维护一个栈，字节码对应的汇编指令就是对栈的各类操作。我们的思路是：对变量的修改操作可以对应为具体的操作码序列，比如对map的修改，可以总结为SHA3+SLOAD。我们插装以太坊节点以获取SLOAD具体对应的操作数，就可以获取到此次交易map的改变量。

* 1. 对数据进行分析。

1. 关键技术点：

代码插装、符号执行、数据分析。

1. 遇到的问题：

1）定位map 2)数据保证 3）误报解决

1. 其他

**项目二套利检测总结：**

1. 项目意义：针对以太坊上的安全漏洞，特别是以太坊上的DAPP即区块链应用进行安全检测。DAPP涉及到多个的智能合约，现有的很多攻击是很多合约共同作用的结果，现有方法无法满足。我们提出了新的方法来对DAPP进行检测。
2. 项目流程：
3. 项目代码框架：1）获取
4. 关键技术点：
5. 遇到的问题：
6. 其他

**项目三 代码优化总结：**

1. 项目意义：针对以太坊上的安全漏洞，特别是以太坊上的DAPP即区块链应用进行安全检测。DAPP涉及到多个的智能合约，现有的很多攻击是很多合约共同作用的结果，现有方法无法满足。我们提出了新的方法来对DAPP进行检测。
2. 项目流程：
3. 项目代码框架：1.插装代码。 2.对数据进行分析。
4. 关键技术点：五元组来保证操作码不变 (gs，pc，st，mem，storage)
5. 遇到的问题：
6. 其他

**项目四 麻将游戏总结：**

1. 项目意义：
2. 项目流程：初始化数据库，创建连接池。启动账户服务器。账户服务器使用express模块监听客户端请求，然后通过send返回JSON后的结果（主要是get请求）。
3. 项目代码框架：1.插装代码。 2.对数据进行分析。
4. 数据库表结构：

表一：账户表：存id和密码；表二：每一次游戏表（游戏动作相关）表三：message：存通讯信息相关（type,message,version）表四：存一局四个用户的得分以及基础信息。表五：用户表：id、昵称、性别、创建的房间等信息。

1. 数据库连接相关：mysql.createpool()；pool.createconnetcion; conn.query()
2. 用户登录相关，实现了游客用户登录。用户登录时，通过点击按钮，调用了客户端的登录函数，首先本地获取缓存的账号，如果有则发送http请求然后通过回调函数返回客户端。客户端处理回调函数，判断是否登录成果，以及用户是否在房间内，以及房间是否存在。如果没有缓存，则创建用户（账号用Date.Now+随机数）。然后登录成功进入大厅。
3. 大厅相关：用户在大厅时，可以选择创建或者加入房间。（此时和XX服务器建立了连接）。创建房间时，用户通过UI组件选择游玩模式以及相关参数（比如局数、底数等）然后向大厅服务器发送创建房间请求。大厅服务器查询数据库，判断用户是否在房间中，然后与游戏服务器通讯，由游戏服务器创建对应游戏模式的房间，并为每个房间设置一个验证的token。返回大厅服务器相关房间信息和游戏服务器信息再返回给用户，然后客户端和游戏服务器建立连接，完成登录。
4. 进入游戏房间的逻辑：两种方式进入：（1）创建后自动进入（2）客户端输入房间号加入；客户端和游戏服务器使用socket.io库实现websocket连接。然后根据用户信息中的roomid,查询房间相关信息放回客户端，并广播给其他客户端。每个socket绑定一个用户id防止重复登录。建立连接后，该连接获取当前房间的信息{该信息存在于服务器代码中}；然后广播该用户的登录信息给其他客户端。每个用户成功进入房间后，服务器都会调用服务器的setread（roomid）函数，当最后一个进入时，当前房间由游戏服务器开始游戏。
5. 游戏主体逻辑：游戏服务器获取当前房间相关信息，然后建立一个结构体存这局游戏的牌库，出牌情况，轮转情况等。然后四个socket分别监听四个客户端吗，比如开始游戏是由最后一个用户加入房间后，该客户端对应的服务器socket监听到setready后，开始初始化游戏。然后要求用户出牌。（特定的客户端操作请求会转化为socket事件发送，每个socket从服务器获取这一局的信息，判断是否执行响应的操作。然后去调用游戏服务器具体的操作函数）
6. 关键技术点：
7. nodejs 回调实现并发：**2）**nodejs的express框架：http协议，函数使用了回调函数的两个参数（request和response对象分别处理http的请求和响应）**3）**nodejs的socket.io库。当用户进入房间开始游戏时，建立一个socket连接通道，监听客户端。
8. 遇到的问题：
9. 其他
10. **游戏中的断线重连？**客户端用户进入大厅从大厅服务器获取用户信息时，会判断返还的roomid是否存在，如果存在则先向大厅服务器发起加入房间的请求。大厅服务器从数据库获取用户信息，然后向游戏服务器发送http请求查询房间是否还存在，然后安排用户进入房间。
11. **游戏的消息通信过程：**

socket.js文件存有socket连接的map,key是用户id,发送消息时：获取到该socket连接，然后使用socket.emit就可以由游戏服务器向指定客户端发送消息，传递信息。

1. **服务器（以及客户端）的部署：**

**项目五 Go聊天室套总结：**

1. 项目意义：
2. 项目流程：
3. 项目代码框架：
4. 关键技术点：
5. 遇到的问题：
6. 其他

**十大排序思想：**

1. **选择排序：**每次选择最小的一个数
2. **插入排序：**前段有序，每次选一个插入到合适位置
3. **冒泡排序：**两两交换，内层循环将最大值移到最右端
4. **归并排序：**分冶法，每次分两半处理，然后合并（选最小的存入辅助数组）
5. **快速排序：**每次选一个元素作为中轴分两半，左边小，右边大。再对左段和右端分别排序。

**tail echo ps grep kill touch mkdir cd su mv rm cp tar zip whereis chmod cat ls netstat ifconfig reboot export source lsof**

**黑盒测试的方法：**

（1）等价类（2）边界值（3）判定表法（3）输入输出覆盖法（4）经验法（5）流程图法

**白盒测试的方法：**

**Linux:**

ps aux ; ps elf ; ps aT;top

&加在命令行后可以使命令在后台执行。ctrl+z挂起当前进程；fg和bg；jobs或ps + kill;

ls -lh

定时重启：crontab -a进入 启动编辑页面。start\_minut end\_hour \* \* \* root reboot; enable crond.service + restart crond.service

修改文件路径的三种方法：(1)export PATH=path:$PATH (2)将该语句加入~/.bashrc ,并用source起效（3）该语句加入/etc/profile，并用source起效。（4）/etc/environment中修改路径。

查看端口：lsof -i : 端口号；netstat -tunlp | grep 端口号

24号面试重点：

（1）内存泄漏：定义：为对象动态申请了内存，使用结束后没有释放，导致了内存得不到释放可能会导致程序崩溃。分为:有GC的内存泄漏和没有GC的内存泄漏，还有操作系统层面的泄漏。没有GC内存泄漏：new了后没有delete，检测方法：源码分析，日志分析，工具查看内存使用情况。有GC的内存泄漏：原因：对象不在使用，但是被引用（被实例占有）无法被垃圾回收。产生原因：静态变量（特别是集合类型，单引用结束后，被总的引用）；资源链接未关闭（数据库、网络、文件I/O等）。linux内存泄漏问题解决：先通过ps -elf查询进程pid，然后通过top查看内存使用情况，或者ps -aux | grep pid,或者cat /proc/pid/status 或者free查看总体情况；

**扩展1：**内存溢出原因1：内存过小2：内存泄漏导致的内存紧缺3：一次性数据量过大4：死循环。

**扩展2：**Go语言垃圾回收：三色标记，白灰黑，并发，写屏障；python垃圾回收：引用计数+标记清除（解决循环引用；存活/死亡容器；例子c,d 2,2->1,1->0,0->死亡容器->查看存活容器）+分代收集（提高效率，标记清楚存活的代数+1,代数越大越不执行回收）；Java垃圾回收：分为新生代+老年代+永久代，对象先到eden,eden满时，minorGC,然后survive0和survive1区交替复制清除；超过阈值送到老年代区，老年代满了执行full GC然后进行标记整理算法。永久代存储元数据，java类方法等

**扩展3：java中的引用类型。**强引用(无法回收)；软引用（内存溢出前再回收）；弱引用（垃圾回收自动回收）；虚引用（唯一用处，被收集到时会被通知）

1. sql语句

修改表alter table mytable add col char(20); alter table mytable drop column col;

更新表： UPDATE mytable set col = val where id =1;

增加行：（1）全列插入：INSERT INTO mytable values(a,b,c,d);（2）部份列插入：INSERT INTO mytable(col1,col2,col3) values(a,b,c);

修改行：UPDATE mytable set col =val where id = 1;

范围查询：SELECT col FROM mytable where id between 3 and 5; where id in(3,4,5)

排序：SELECT col FROM mytable where sex = 1 order by score asc; order by score desc;

其他函数：SELECT sum(score)/count(\*) from mytable where sex = 1; max(id) ; min(id)

分组： SELECT id count(id) from mytable group by id;( 统计每类id各有多少种)

（3）自我评价：；优缺点总结：；最大成就和挫折：；

（4）APP登录测试

1.APP核心功能的流程：打开APP,选择XX模块，选择XX功能，确认XX功能，结算XX，反馈。功能+性能+兼容性+安全性

**扩展：（1）web测试和APP测试区别：**功能测试方面，web是浏览器模式，app是客户端模式。客户端模式要考虑安装升级卸载等功能。性能上web主要考虑网络响应，app在此基础上还要集合手机（耗电，cpu，内存，甚至是发热情况）来考虑性能。还有兼容性等的考虑。

（5）测试案例

（6）快排

func partition(list []int ,l,r int)int{

pivot :=list[l]

i,j:=l+1,r

for i<=j{

for i<r&&list[i]<=pivot{

i++

}

for j>l&&list[j]>pivot{

j--

}

if i>=j{

break

}

list[i],list[j] = list[j],list[i]

}

list[l],list[j] = list[j],list[l]

return j

}

func quickSort(list []int,l,r int){

if l<r{

mid :=partition2(list,l,r)

quickSort(list,l,mid-1)

quickSort(list,mid+1,r)

}

}