1. Java 基础知识点
2. Java 常见集合
3. 高并发编程（JUC 包）
4. JVM 内存管理
5. Java 8 知识点
6. 网络协议相关
7. 数据库相关
8. MVC 框架相关
9. 大数据相关
10. Linux 命令相关

**Java 基础知识**

**01. 面向对象的特性有哪些？**

**答：**封装、继承和多态。

**02. Java 中覆盖和重载是什么意思？**

**解析：**覆盖和重载是比较重要的基础知识点，并且容易混淆，所以面试中常见。

答：覆盖（Override）是指子类对父类方法的一种重写，只能比父类抛出更少的异常，访问权限不能比父类的小。

被覆盖的方法不能是 private 的，否则只是在子类中重新定义了一个方法；重载（Overload）表示同一个类中可以有多个名称相同的方法，但这些方法的参数列表各不相同。

**面试官：** 那么构成重载的条件有哪些？

**答：**参数类型不同、参数个数不同、参数顺序不同。

**面试官：** 函数的返回值不同可以构成重载吗？为什么？

**答：**不可以，因为 Java 中调用函数并不需要强制赋值。举例如下：

如下两个方法：

　　　　void f(){}　　　　int f(){ return 1;}

只要编译器可以根据语境明确判断出语义，比如在 int x = f();中，那么的确可以据此区分重载方法。

不过， 有时你并不关心方法的返回值，你想要的是方法调用的其他效果 （这常被称为 “为了副作用而调用”），这时你可能会调用方法而忽略其返回值，所以如果像下面的调用：

　　　　fun();

此时 Java 如何才能判断调用的是哪一个 f( ) 呢？别人如何理解这种代码呢？所以，根据方法返回值来区分重载方法是行不通的。

**03. 抽象类和接口的区别有哪些？**

**答：**

1. 抽象类中可以没有抽象方法；接口中的方法必须是抽象方法；
2. 抽象类中可以有普通的成员变量；接口中的变量必须是 static final 类型的，必须被初始化 , 接口中只有常量，没有变量。
3. 抽象类只能单继承，接口可以继承多个父接口；
4. Java8 中接口中会有 default 方法，即方法可以被实现。

**面试官：**抽象类和接口如何选择？

**答：**

1. 如果要创建不带任何方法定义和成员变量的基类，那么就应该选择接口而不是抽象类。
2. 如果知道某个类应该是基类，那么第一个选择的应该是让它成为一个接口，只有在必须要有方法定义和成员变量的时候，才应该选择抽象类。

因为抽象类中允许存在一个或多个被具体实现的方法，只要方法没有被全部实现该类就仍是抽象类。

**04. Java 和 C++ 的区别：**

**解析：**虽然我们不太懂 C++，但是就是会这么问，尤其是三面（总监级别）面试中。

**答：**

1. 都是面向对象的语言，都支持封装、继承和多态；
2. **指针：**Java 不提供指针来直接访问内存，程序更加安全；
3. **继承：** Java 的类是单继承的，C++ 支持多重继承； Java 通过一个类实现多个接口来实现 C++ 中的多重继承； Java 中类不可以多继承，但是！！！接口可以多继承；
4. **内存：** Java 有自动内存管理机制，不需要程序员手动释放无用内存。

**05. Java 中的值传递和引用传递**

**答：**

值传递是指对象被值传递，意味着传递了对象的一个副本，即使副本被改变，也不会影响源对象。引用传递是指对象被引用传递，意味着传递的并不是实际的对象，而是对象的引用。

因此，外部对引用对象的改变会反映到所有的对象上。

**06. JDK 中常用的包有哪些？**

**答：**java.lang、java.util、java.io、java.net、java.sql。

**07. JDK，JRE 和 JVM 的联系和区别：**

**答：**

**JDK**是 java 开发工具包，是 java 开发环境的核心组件，并提供编译、调试和运行一个 java 程序所需要的所有工具，可执行文件和二进制文件，是一个平台特定的软件。

**JRE**是 java 运行时环境，是 JVM 的实施实现，提供了运行 java 程序的平台。JRE 包含了 JVM，但是不包含 java 编译器 / 调试器之类的开发工具。

**JVM**是 java 虚拟机，当我们运行一个程序时，JVM 负责将字节码转换为特定机器代码，JVM 提供了内存管理 / 垃圾回收和安全机制等。

这种独立于硬件和操作系统，正是 java 程序可以一次编写多处执行的原因。

**区别：**

1. JDK 用于开发，JRE 用于运行 java 程序；
2. JDK 和 JRE 中都包含 JVM；
3. JVM 是 java 编程语言的核心并且具有平台独立性。

小结：本节主要阐述了 Java 基础知识点，这些问题主要是一面面试官在考察，难度不大，适当复习下，应该没什么问题。

**Java 中常见集合**

集合这方面的考察相当多，这部分是面试中必考的知识点。

**01. 说说常见的集合有哪些吧？**

**答：**

Map 接口和 Collection 接口是所有集合框架的父接口：

1. Collection 接口的子接口包括：Set 接口和 List 接口；

2. Map 接口的实现类主要有：HashMap、TreeMap、Hashtable、ConcurrentHashMap 以及 Properties 等；

3. Set 接口的实现类主要有：HashSet、TreeSet、LinkedHashSet 等；

4. List 接口的实现类主要有：ArrayList、LinkedList、Stack 以及 Vector 等。

**02. HashMap 和 Hashtable 的区别有哪些？（必问）**

**答：**

1. HashMap 没有考虑同步，是线程不安全的；Hashtable 使用了 synchronized 关键字，是线程安全的；
2. 前者允许 null 作为 Key；后者不允许 null 作为 Key。

**03. HashMap 的底层实现你知道吗？**

**答：**

在 Java8 之前，其底层实现是数组 + 链表实现，Java8 使用了数组 + 链表 + 红黑树实现。此时你可以简单的在纸上画图分析：

**04. ConcurrentHashMap 和 Hashtable 的区别？ （必问）**

**答：**

ConcurrentHashMap 结合了 HashMap 和 HashTable 二者的优势。

HashMap 没有考虑同步，hashtable 考虑了同步的问题。但是 hashtable 在每次同步执行时都要锁住整个结构。

ConcurrentHashMap 锁的方式是稍微细粒度的。 ConcurrentHashMap 将 hash 表分为 16 个桶（默认值），诸如 get,put,remove 等常用操作只锁当前需要用到的桶。

**面试官：**ConcurrentHashMap 的具体实现知道吗？

**答：**

1. 该类包含两个静态内部类 HashEntry 和 Segment；前者用来封装映射表的键值对，后者用来充当锁的角色；
2. Segment 是一种可重入的锁 ReentrantLock，每个 Segment 守护一个 HashEntry 数组里得元素，当对 HashEntry 数组的数据进行修改时，必须首先获得对应的 Segment 锁。

**05. HashMap 的长度为什么是 2 的幂次方？**

**答：**

1. 通过将 Key 的 hash 值与 length-1 进行 & 运算，实现了当前 Key 的定位，2 的幂次方可以减少冲突（碰撞）的次数，提高 HashMap 查询效率；
2. 如果 length 为 2 的次幂  则 length-1 转化为二进制必定是 11111……的形式，在于 h 的二进制与操作效率会非常的快，而且空间不浪费；
3. 如果 length 不是 2 的次幂，比如 length 为 15，则 length-1 为 14，对应的二进制为 1110，在于 h 与操作，最后一位都为 0，而 0001，0011，0101，1001，1011，0111，1101 这几个位置永远都不能存放元素了，空间浪费相当大。

更糟的是这种情况中，数组可以使用的位置比数组长度小了很多，这意味着进一步增加了碰撞的几率，减慢了查询的效率！这样就会造成空间的浪费。

**06. List 和 Set 的区别是啥？**

**答：**List 元素是有序的，可以重复；Set 元素是无序的，不可以重复。

**07. List、Set 和 Map 的初始容量和加载因子**

**答：**

**1. List**

* ArrayList 的初始容量是 10；加载因子为 0.5； 扩容增量：原容量的 0.5 倍 +1；一次扩容后长度为 16。
* Vector 初始容量为 10，加载因子是 1。扩容增量：原容量的 1 倍，如 Vector 的容量为 10，一次扩容后是容量为 20。

**2. Set**

HashSet，初始容量为 16，加载因子为 0.75； 扩容增量：原容量的 1 倍； 如 HashSet 的容量为 16，一次扩容后容量为 32

**3. Map**

HashMap，初始容量 16，加载因子为 0.75； 扩容增量：原容量的 1 倍； 如 HashMap 的容量为 16，一次扩容后容量为 32

**08. Comparable 接口和 Comparator 接口有什么区别？**

**答：**

1. 前者简单，但是如果需要重新定义比较类型时，需要修改源代码。
2. 后者不需要修改源代码，自定义一个比较器，实现自定义的比较方法。

**09. Java 集合的快速失败机制 “fail-fast”**

**答：**

它是 java 集合的一种错误检测机制，当多个线程对集合进行结构上的改变的操作时，有可能会产生 fail-fast 机制。

**例如 ：**假设存在两个线程（线程 1、线程 2），线程 1 通过 Iterator 在遍历集合 A 中的元素，在某个时候线程 2 修改了集合 A 的结构（是结构上面的修改，而不是简单的修改集合元素的内容），那么这个时候程序就会抛出 ConcurrentModificationException 异常，从而产生 fail-fast 机制。

**原因：** 迭代器在遍历时直接访问集合中的内容，并且在遍历过程中使用一个 modCount 变量。集合在被遍历期间如果内容发生变化，就会改变 modCount 的值。

每当迭代器使用 hashNext()/next() 遍历下一个元素之前，都会检测 modCount 变量是否为 expectedmodCount 值，是的话就返回遍历；否则抛出异常，终止遍历。

**解决办法：**

1. 在遍历过程中，所有涉及到改变 modCount 值得地方全部加上 synchronized；
2. 使用 CopyOnWriteArrayList 来替换 ArrayList。

小结：本小节是 Java 中关于集合的考察，是 Java 岗位面试中必考的知识点，除了应该掌握以上的问题，包括各个集合的底层实现也建议各位同学阅读，加深理解。

**高并发编程**

在 Java 5.0 提供了 java.util.concurrent（简称 JUC ）包，在此包中增加了在并发编程中很常用的实用工具类，用于定义类似于线程的自定义子系统，包括线程池、异步 IO 和轻量级任务框架。

**01. 多线程和单线程的区别和联系：**

**答：**

1. 在单核 CPU 中，将 CPU 分为很小的时间片，在每一时刻只能有一个线程在执行，是一种微观上轮流占用 CPU 的机制。
2. 多线程会存在线程上下文切换，会导致程序执行速度变慢，即采用一个拥有两个线程的进程执行所需要的时间比一个线程的进程执行两次所需要的时间要多一些。

**结论：**即采用多线程不会提高程序的执行速度，反而会降低速度，但是对于用户来说，可以减少用户的响应时间。

**02. 如何指定多个线程的执行顺序？**

**解析：**面试官会给你举个例子，如何让 10 个线程按照顺序打印 0123456789？（写代码实现）

**答：**

1. 设定一个 orderNum，每个线程执行结束之后，更新 orderNum，指明下一个要执行的线程。并且唤醒所有的等待线程。
2. 在每一个线程的开始，要 while 判断 orderNum 是否等于自己的要求值！！不是，则 wait，是则执行本线程。

**03. 线程和进程的区别（必考）**

**答：**

1. 进程是一个 “执行中的程序”，是系统进行资源分配和调度的一个独立单位；
2. 线程是进程的一个实体，一个进程中拥有多个线程，线程之间共享地址空间和其它资源（所以通信和同步等操作线程比进程更加容易）；
3. 线程上下文的切换比进程上下文切换要快很多。
   * （1）进程切换时，涉及到当前进程的 CPU 环境的保存和新被调度运行进程的 CPU 环境的设置。
   * （2）线程切换仅需要保存和设置少量的寄存器内容，不涉及存储管理方面的操作。

**04. 多线程产生死锁的 4 个必要条件？**

**答：**

1. 互斥条件：一个资源每次只能被一个线程使用；
2. 请求与保持条件：一个线程因请求资源而阻塞时，对已获得的资源保持不放；
3. 不剥夺条件：进程已经获得的资源，在未使用完之前，不能强行剥夺；
4. 循环等待条件：若干线程之间形成一种头尾相接的循环等待资源关系。

面试官：如何避免死锁？（经常接着问这个问题哦~）

答：指定获取锁的顺序，举例如下：

1. 比如某个线程只有获得 A 锁和 B 锁才能对某资源进行操作，在多线程条件下，如何避免死锁？
2. 获得锁的顺序是一定的，比如规定，只有获得 A 锁的线程才有资格获取 B 锁，按顺序获取锁就可以避免死锁！！！

**05. sleep( ) 和 wait( n)、wait( ) 的区别：**

**答：**

1. sleep 方法：是 Thread 类的静态方法，当前线程将睡眠 n 毫秒，线程进入阻塞状态。当睡眠时间到了，会解除阻塞，进行可运行状态，等待 CPU 的到来。睡眠不释放锁（如果有的话）；
2. wait 方法：是 Object 的方法，必须与 synchronized 关键字一起使用，线程进入阻塞状态，当 notify 或者 notifyall 被调用后，会解除阻塞。但是，只有重新占用互斥锁之后才会进入可运行状态。睡眠时，释放互斥锁。

**06. synchronized 关键字：**

**答：**

底层实现：

1. 进入时，执行 monitorenter，将计数器 +1，释放锁 monitorexit 时，计数器-1；
2. 当一个线程判断到计数器为 0 时，则当前锁空闲，可以占用；反之，当前线程进入等待状态。

**含义：（monitor 机制）**

Synchronized 是在加锁，加对象锁。对象锁是一种重量锁（monitor），synchronized 的锁机制会根据线程竞争情况在运行时会有偏向锁（单一线程）、轻量锁（多个线程访问 synchronized 区域）、对象锁（重量锁，多个线程存在竞争的情况）、自旋锁等。

该关键字是一个几种锁的封装。

**07. volatile 关键字**

**解析：**关于指令重排序的问题，可以查阅 DCL 双检锁失效相关资料。

**答：**

该关键字可以保证可见性不保证原子性。

**功能：**

1. 主内存和工作内存，直接与主内存产生交互，进行读写操作，保证可见性；
2. 禁止 JVM 进行的指令重排序。

**08. ThreadLocal（线程局部变量）关键字：**

**答：**

当使用 ThreadLocal 维护变量时，其为每个使用该变量的线程提供独立的变量副本，所以每一个线程都可以独立的改变自己的副本，而不会影响其他线程对应的副本。

ThreadLocal 内部实现机制：

1. 每个线程内部都会维护一个类似 HashMap 的对象，称为 ThreadLocalMap，里边会包含若干了 Entry（K-V 键值对），相应的线程被称为这些 Entry 的属主线程；
2. Entry 的 Key 是一个 ThreadLocal 实例，Value 是一个线程特有对象。Entry 的作用即是：为其属主线程建立起一个 ThreadLocal 实例与一个线程特有对象之间的对应关系；
3. Entry 对 Key 的引用是弱引用；Entry 对 Value 的引用是强引用。

**09. Atomic 关键字：**

**答：**可以使基本数据类型以原子的方式实现自增自减等操作。

**10. 线程池有了解吗？（必考）**

**答：**

java.util.concurrent.ThreadPoolExecutor 类就是一个线程池。客户端调用 ThreadPoolExecutor.submit(Runnable task) 提交任务，线程池内部维护的工作者线程的数量就是该线程池的线程池大小，有 3 种形态：

* 当前线程池大小 ：表示线程池中实际工作者线程的数量；
* 最大线程池大小 （maxinumPoolSize）：表示线程池中允许存在的工作者线程的数量上限；
* 核心线程大小 （corePoolSize ）：表示一个不大于最大线程池大小的工作者线程数量上限。

1. 如果运行的线程少于 corePoolSize，则 Executor 始终首选添加新的线程，而不进行排队；
2. 如果运行的线程等于或者多于 corePoolSize，则 Executor 始终首选将请求加入队列，而不是添加新线程；
3. 如果无法将请求加入队列，即队列已经满了，则创建新的线程，除非创建此线程超出 maxinumPoolSize， 在这种情况下，任务将被拒绝。

限于篇幅有限，更多高并发编程中的问题，请参考：

1.  Java 多线程编程核心技术

2. Java多线程与并发编程

小结：本小节内容涉及到 Java 中多线程编程，线程安全等知识，是面试中的重点和难点。

**JVM 内存管理**

既然是 Java 开发面试，那么对 JVM 的考察当然也是必须的，面试官一般会问你对 JVM 有了解吗？

我通常都会把我所了解的都说一遍，包括：JVM 内存划分、JVM 垃圾回收的含义，有哪些 GC 算法，年轻代和老年代各自的特点统统阐述一遍。

**01. JVM 内存划分：**

1. 方法区（线程共享）：常量、静态变量、JIT(即时编译器) 编译后的代码也都在方法区；
2. 堆内存（线程共享）：垃圾回收的主要场所；
3. 程序计数器： 当前线程执行的字节码的位置指示器；
4. 虚拟机栈（栈内存）：保存局部变量、基本数据类型变量以及堆内存中某个对象的引用变量；
5. 本地方法栈 ：为 JVM 提供使用 native 方法的服务。

**02. 类似-Xms、-Xmn 这些参数的含义：**

**答：**

**堆内存分配：**

1. JVM 初始分配的内存由-Xms 指定，默认是物理内存的 1/64；
2. JVM 最大分配的内存由-Xmx 指定，默认是物理内存的 1/4；
3. 默认空余堆内存小于 40% 时，JVM 就会增大堆直到-Xmx 的最大限制；空余堆内存大于 70% 时，JVM 会减少堆直到 -Xms 的最小限制；
4. 因此服务器一般设置-Xms、-Xmx 相等以避免在每次 GC 后调整堆的大小。对象的堆内存由称为垃圾回收器的自动内存管理系统回收。

**非堆内存分配：**

1. JVM 使用-XX:PermSize 设置非堆内存初始值，默认是物理内存的 1/64；
2. 由 XX:MaxPermSize 设置最大非堆内存的大小，默认是物理内存的 1/4；
3. -Xmn2G：设置年轻代大小为 2G；
4. -XX:SurvivorRatio，设置年轻代中 Eden 区与 Survivor 区的比值。

**03. 垃圾回收算法有哪些？**

**答：**

1. 引用计数 ：原理是此对象有一个引用，即增加一个计数，删除一个引用则减少一个计数。垃圾回收时，只用收集计数为 0 的对象。此算法最致命的是无法处理循环引用的问题；
2. 标记-清除 ：此算法执行分两阶段。第一阶段从引用根节点开始标记所有被引用的对象，第二阶段遍历整个堆，把未标记的对象清除；

此算法需要暂停整个应用，同时，会产生内存碎片；

1. 复制算法 ：此算法把内存空间划为两个相等的区域，每次只使用其中一个区域。垃圾回收时，遍历当前使用区域，把正在使用中的对象复制到另外一个区域中；

此算法每次只处理正在使用中的对象，因此复制成本比较小，同时复制过去以后还能进行相应的内存整理，不会出现 “碎片” 问题。当然，此算法的缺点也是很明显的，就是需要两倍内存空间；

1. 标记-整理 ：此算法结合了 “标记-清除” 和 “复制” 两个算法的优点。也是分两阶段，第一阶段从根节点开始标记所有被引用对象，第二阶段遍历整个堆，把清除未标记对象并且把存活对象 “压缩” 到堆的其中一块，按顺序排放。

此算法避免了 “标记-清除” 的碎片问题，同时也避免了 “复制” 算法的空间问题。

**04. root 搜索算法中，哪些可以作为 root？**

**答：**

* 被启动类（bootstrap 加载器）加载的类和创建的对象；
* JavaStack 中的引用的对象 (栈内存中引用的对象)；
* 方法区中静态引用指向的对象；
* 方法区中常量引用指向的对象；
* Native 方法中 JNI 引用的对象。

**05. GC 什么时候开始？**

**答：**

GC 经常发生的区域是堆区，堆区还可以细分为新生代、老年代，新生代还分为一个 Eden 区和两个 Survivor 区。

1. 对象优先在 Eden 中分配，当 Eden 中没有足够空间时，虚拟机将发生一次 Minor GC，因为 Java 大多数对象都是朝生夕灭，所以 Minor GC 非常频繁，而且速度也很快；
2. Full GC，发生在老年代的 GC，当老年代没有足够的空间时即发生 Full GC，发生 Full GC 一般都会有一次 Minor GC。

大对象直接进入老年代，如很长的字符串数组，虚拟机提供一个；XX:PretenureSizeThreadhold 参数，令大于这个参数值的对象直接在老年代中分配，避免在 Eden 区和两个 Survivor 区发生大量的内存拷贝；

1. 发生 Minor GC 时，虚拟机会检测之前每次晋升到老年代的平均大小是否大于老年代的剩余空间大小，如果大于，则进行一次 Full GC，如果小于，则查看 HandlePromotionFailure 设置是否允许担保失败，如果允许，那只会进行一次 Minor GC，如果不允许，则改为进行一次 Full GC。

**06. 内存泄漏和内存溢出**

**答：**

**概念：**

1. 内存溢出指的是内存不够用了；
2. 内存泄漏是指对象可达，但是没用了。即本该被 GC 回收的对象并没有被回收；
3. 内存泄露是导致内存溢出的原因之一；内存泄露积累起来将导致内存溢出。

**内存泄漏的原因分析：**

1. 长生命周期的对象引用短生命周期的对象；
2. 没有将无用对象置为 null。

小结：本小节涉及到 JVM 虚拟机，包括对内存的管理等知识，相对较深。除了以上问题，面试官会继续问你一些比较深的问题，可能也是为了看看你的极限在哪里吧。

比如：内存调优、内存管理，是否遇到过内存泄漏的实际案例、是否真正关心过内存等。由于本人实际项目经验不足，这些深层次问题并没有接触过，各位有需要可以上网查阅。

**Java 8 相关知识**

关于 Java8 中新知识点，面试官会让你说说 Java8 你了解多少，下边主要阐述我所了解，并且在面试中回答的 Java8 新增知识点。

**0.1**HashMap 的底层实现有变化：HashMap 是数组 + 链表 + 红黑树（JDK1.8 增加了红黑树部分）实现。

**02.**JVM 内存管理方面，由元空间代替了永久代。

**区别：**

1. 元空间并不在虚拟机中，而是使用本地内存；

2. 默认情况下，元空间的大小仅受本地内存限制；

3. 也可以通过 -XX：MetaspaceSize 指定元空间大小。

**03.**Lambda 表达式（也称为闭包），允许我们将函数当成参数传递给某个方法，或者把代码本身当做数据处理。

**04.**函数式接口：指的是只有一个函数的接口，java.lang.Runnable 和 java.util.concurrent.Callable 就是函数式接口的例子；java8 提供了一个特殊的注解 @Functionallnterface 来标明该接口是一个函数式接口。

**05.**引入重复注解：Java 8 中使用 @Repeatable 注解定义重复注解。

**06.**接口中可以实现方法 default 方法。

**07.**注解的使用场景拓宽： 注解几乎可以使用在任何元素上：局部变量、接口类型、超类和接口实现类，甚至可以用在函数的异常定义上。

**08.** 新的包 java.time 包

1. 包含了所有关于日期、时间、时区、持续时间和时钟操作的类。
2. 这些类都是不可变的、线程安全的。

小结：Java8 的一些新特性，面试官一般情况下不要求你有多么精通，主要是看看你有没有一些了解。

**网络协议相关**

网络协议方面，考察最多的包括服务器和客户端在三次握手、四次挥手过程中的状态变化；还有网络拥塞控制，及其解决办法等。

**01. 三次握手、四次挥手示意图：**

总共有四种状态：主动建立连接、主动断开连接、被动建立连和被动断开连接

**两两组合还是 4 种组合：**

1. 主动建立连接、主动断开连接会经历的状态：  
   SYNC\_SENT——ESTABLISHED—-FIN\_WAIT\_1—-FIN\_WAIT\_2—-TIME\_WAIT
2. 主动建立连接、被动断开连接会经历的状态：  
   SYNC\_SENT——ESTABLISHED—-CLOSE\_WAIT—-LAST\_ACK
3. 被动建立连接、主动断开连接会经历的状态：  
   LISTEN—-SYN\_RCVD—-ESTABLISHED—-FIN\_WAIT\_1—-FIN\_WAIT\_2—-TIME\_WAIT
4. 被动建立连接、被动断开连接会经历的状态：  
   LISTEN—-SYN\_RCVD—-ESTABLISHED—-CLOSE\_WAIT—-LAST\_ACK

**02. 滑动窗口机制**

由发送方和接收方在三次握手阶段，互相将自己的最大可接收的数据量告诉对方。

也就是自己的数据接收缓冲池的大小。这样对方可以根据已发送的数据量来计算是否可以接着发送。

在处理过程中，当接收缓冲池的大小发生变化时，要给对方发送更新窗口大小的通知。

**03. 拥塞避免机制**

**拥塞：**对资源的需求超过了可用的资源。若网络中许多资源同时供应不足，网络的性能就要明显变坏，整个网络的吞吐量随之负荷的增大而下降。

**拥塞控制：**防止过多的数据注入到网络中，使得网络中的路由器或链路不致过载。

**拥塞控制方法：**

* 慢开始 + 拥塞避免；
* 快重传 + 快恢复。

**04. 浏览器中输入：“www.xxx.com” 之后都发生了什么？请详细阐述。**

**解析：**经典的网络协议问题。

**答：**

1. 由域名→IP 地址  
   寻找 IP 地址的过程依次经过了浏览器缓存、系统缓存、hosts 文件、路由器缓存、 递归搜索根域名服务器。
2. 建立 TCP/IP 连接（三次握手具体过程）
3. 由浏览器发送一个 HTTP 请求
4. 经过路由器的转发，通过服务器的防火墙，该 HTTP 请求到达了服务器
5. 服务器处理该 HTTP 请求，返回一个 HTML 文件
6. 浏览器解析该 HTML 文件，并且显示在浏览器端
7. 这里需要注意：
   * HTTP 协议是一种基于 TCP/IP 的应用层协议，进行 HTTP 数据请求必须先建立 TCP/IP 连接
   * 可以这样理解：HTTP 是轿车，提供了封装或者显示数据的具体形式；Socket 是发动机，提供了网络通信的能力。
   * 两个计算机之间的交流无非是两个端口之间的数据通信 , 具体的数据会以什么样的形式展现是以不同的应用层协议来定义的。

**05. 常见 HTTP 状态码**

1. 1xx（临时响应）
2. 2xx（成功）
3. 3xx（重定向）：表示要完成请求需要进一步操作
4. 4xx（错误）：表示请求可能出错，妨碍了服务器的处理
5. 5xx（服务器错误）：表示服务器在尝试处理请求时发生内部错误
6. 常见状态码：
   * 200（成功）
   * 304（未修改）：自从上次请求后，请求的网页未修改过。服务器返回此响应时，不会返回网页内容
   * 401（未授权）：请求要求身份验证
   * 403（禁止）：服务器拒绝请求
   * 404（未找到）：服务器找不到请求的网页

**06. TCP 和 UDP 的区别：**

**答：**

1. 回答发送数据前是否存在建立连接的过程；
2. ＴＣＰ过确认机制，丢包可以重发，保证数据的正确性；ＵＤＰ不保证正确性，只是单纯的负责发送数据包；
3. UDP 是面向报文的。发送方的 UDP 对应用程序交下来的报文，在添加首部后就向下交付给 IP 层。既不拆分，也不合并，而是保留这些报文的边界，因 此，应用程序需要选择合适的报文大小；
4. UDP 的头部，只有 8 个字节，相对于 TCP 头部的 20 个字节信息包的额外开销很小。