

# 《网络科学导论》

**课程报告**

# 姓 名： 陈聪 学 号：20181003989

# 院 系：计算机学院 专 业： 网络工程

指导老师： 李振华 职 称： 教授

网络科学导论课作业

问题一:在计算机网络领域，你最感兴趣的一个问题？

答：在我小时候第一次接触电脑的时，就对电脑产生疑惑，这些游戏是从哪来的，这些电影电视是怎么传到电脑上的，当时的我仅仅只知道有个叫做互联网的东西，但一直没有深入了解，所以现如今让我十分想搞清楚的一个问题就是：物联网到底是什么？

物联网（英语：Internet of Things，缩写IoT）是互联网、传统电信网等信息承载体，让所有能行使独立功能的普通物体实现互联互通的网络。物联网是新一代信息技术的重要组成部分，也是“信息化”时代的重要发展阶段。物联网就是“物物相连的互联网”。这有两层意思：其一，物联网的核心和基础仍然是互联网，是在互联网基础上的延伸和扩展的网络；其二，其用户端延伸和扩展到了任何物品与物品之间，进行信息交换和通信，也就是物物相息。

物联网将现实世界数位化，应用范围十分广泛。在物联网上，每个人都可以应用电子标签将真实的物体上网联结，在物联网上都可以查出它们的具体位置。通过物联网可以用中心计算机对机器、设备、人员进行集中管理、控制，也可以对家庭设备、汽车进行遥控，以及搜索位置、防止物品被盗等，类似自动化操控系统，同时通过收集这些小事的数据，最后可以聚集成大数据，包含重新设计道路以减少车祸、都市更新、灾害预测与犯罪防治、流行病控制等等社会的重大改变。

问题二：计算机网络领域，你最感兴趣的企业家或科学家是谁？为其列一个传记；或者选择当代网络的一个热点事件，谈一谈这一事件的启示。

答：他是计算机先驱之一，他开发了程序设计的框架结构。 Edsger Wybe Dijkstra于1930年5月11日生于Rotterdam，，他的父亲，Douwe Wybe Dijkstra是一位化学家，他的母亲，Brechtje Cornelia Kruyper是一位数学家，这种充满科学气息的家庭背景对于他的职业生涯乃至他的整个人生都有着深刻的影响。Edsger Wybe Dijkstra在当地的Gymnasium Erasmianum读高中，1948年，他考入了Leyden大学。他在联合国从事法律方面的工作时却在怀念在Erasmianum的日子，现在，他选择了数学和物理。

Edsger Wybe Dijkstra在三年之内取得了学士学位，这令他的父亲非常高兴，并在1951年9月同意他去英国参加一个夏季的课程，那是一个由剑桥大学开设的，学习电子计算装置程序设计的课程，讲师是著名的M. V. Wilkes。Edsger Wybe Dijkstra的导师让他给Amsterdamr数学中心计算部门的主管Aad van Wijngaarden写一封信，以确定他的基础知识是否足够他去完成该学业。 在那一年前，Van Wijngaarden曾在剑桥学习过，他很快便回复了，信的内容有两点，一是肯定Dijkstra现有的知识已经足够了，二是请他来Amsterdamr作为一名程序设计人员为自己工作。

对于Dijkstra来说，当时还是一名学生，而他在1951年夏季Wilkes所授的学业成为了他日后职业生涯的基础。 在数学中心的“自由阶段” 在没有任何相关知识的情况下，Dijkstra的程序设计生涯开始于改写突变程序和输入Van Wijngaarden已经写好的程序。Van Wijngaarden允许他这样做，这些程序是为MC第一台计算机ARRA I开发的，由C.S. Scholten和J. Loopstra设计完成。 MC计算部门夜以继日的工作，去解决有关那些在Netherland开发的大的方案中为数众多的难题。例如关于Zeeland州安全问题的DELTA计划。 另外一个较大的工程是Fokker友谊飞机的开发，其机翼振动的计算结果需要MC尽最大的能量。

1953年Gerrit Blaauw加入了MC的队伍。第一台ARRA II 构造完成，由于这台机器的可靠性，Fokker飞机公司又订了一台类似的计算机，叫做FERTA。FERTA的速度是ARRA II 的两倍，而且用一套不同类型的代码。Dijkstra为这些机器都研制开发了软件，也包括其后1956年的ARMAC，那也是为MC开发的最后一台计算机。 在完成了FERTA之后，Gerrit Blaauw去往美国为IBM工作，在那里，他从事IBM7030“Stretch”的开发工作，并最终设计和建造了IBM 360系统。 新的挑战：Electrologica 由于巨大、精于计算的机器的开发以走上正轨，Dijkstra, Scholten 和Loopstra又以完成了下一台计算机的准备工作，1956年，MCmanagement和人寿保险公司Nillmij决定成立一个独立的公司：Electrologica，来经营商业电脑。

1952至1956年间，程序设计经历了一个演变的过程，这部分是由于系统分组的复杂性要求一个更具结构性的操作系统，部分是由于科学、数学上的关于程序设计的态度都提出了一个清楚的关于如何提高工作效率的观点。Dijkstra的Shortest Path [Algorithm](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=105662&ss_c=ssc.citiao.link)是在这方面取得的突出进展，因为这种演变是全球性的，所以，在全世界的推动下，一个科学的[计算机语言](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=664318&ss_c=ssc.citiao.link)基础：ALGOL，不久就诞生了。 1958年，Edsger Dijkstra代表Dutch MC出席了11月在Mainz召开的会议，那是一个定义ALGOL详述的准备会议。1959年12月，Dijkstra给ALGOL60下了这样的定义：“一个奇迹就被这样简单的创造了。”最后，1962年的4月，罗马公约同意了其大部分的详述，同年8月，IFIP，国际[程序设计语言](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=8645833&ss_c=ssc.citiao.link)联盟复查并批准了该报告。 1960年的1月，在ALGOL60被定义之后，数学中心首先在荷兰开设了ALGOL60程序设计语言的课程，接着，1961年在英国的Brighton。这是MC一个新的开端：程序设计教育。

Eds1962年，Edsger Dijkstra在TH Eindhoven任全职教授，虽然在国外已经被认为是计算机科学的主席，但Dijkstra强烈反对这个方法，这主要是由于在专业科学知识上的缺乏。他的位置实际上是一个数学教授，他的学生接受了至少三年彻底的数学教育，经过这样一个时期，他们都能作为信息学方面的专家了。那些[数学训练](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=66804799&ss_c=ssc.citiao.link)是以应用数学原理为基础的。由此，信息学有了适合其学科本身的数学方法。 1967年，Dijkstra陷入了情绪上的危机，他第一个学生的论文被他在Eindhoven数学上的同事拒收了，而这些同事对于计算机科学一直是带有偏见的。对于他和他的妻子来说，那段不景气的日子是他们一生中最困难的时期。但是他很快有恢复了，并开始投入编写：[结构化编程](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=63770194&ss_c=ssc.citiao.link)笔记。 而Dijkstra在Eindhoven的同事对此不是保持沉默，就是完全消极的反应，但Dijkstra选择了正确的还击方式：他给欧洲和美国的同事们复印了20多份稿件。 Burroughs和彻底的自由 1973年，Dijkstra成为了Burroughs的研究员，他减少了在Eindhoven TH的工作。这个决定使他能够去写科学报告，他为Burroughs写了500多篇，还可以如愿的出国旅行。他成了一个[自由人](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=246344&ss_c=ssc.citiao.link)，而且拥有该公司最小的实验室：他的书房。

Dijkstra在旅行途中多次有机会参观在Austin的得克萨斯[州立大学](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=71518266&ss_c=ssc.citiao.link)，在那里他还做过几次讲座。1984年，他有了一个担任那里计算机科学学院的全职教授的机会，他觉得他在那里会感到像家一样，于是他和他的妻子搬到美国居住。这便开始了他15年的教学生活：编写、讨论[程序设计技术](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=10037556&ss_c=ssc.citiao.link)。除此以外，整个美国的好客给他和他的妻子都留下了深刻的印象。 得克萨斯州立大学：超型计算机 1999年，Dijkstra在他69岁的时候，结束了作为教授的职业生涯。回顾47年的艰苦工作，为了一个更好、更简单、更准确的编程方法而不停地努力奋斗，使“符号、说明清楚”。Dijkstra认为最具价值的是他对学生的教导，而且能够向人们展示这项工作可以做的比他们所知或所想象的更加出色。能够做到吸引他人，已经成为了他最有收获的活动。 “对于我来说，计算机科学上的第一个挑战是如何把命令维持在有限个内，然而巨大的、分立的宇宙是复杂地缠绕着的。第二个也是同样重要的挑战是如何传授解决那[第一个问题](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=156161518&ss_c=ssc.citiao.link)的方法：只培养你个人的才智（那会随你进入坟墓的东西）是不够的，你必须教会其他人如何去发挥他们的才智。你越关注这两个挑战，你越会清楚的看到它们只不过是同一枚硬币的两个面：自学是去发现什么东西是可以被教会的。” （"My hopes to Computer Science") Edsger Wybe Dijkstra因患癌症于2002年8月6日在Nuenen, The Netherlands逝世 。

　Dijkstra被西方学术界称为“[结构程序设计](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=85471749&ss_c=ssc.citiao.link)之父”和“先知先觉”（Oracle），他一生致力于把程序设计发展成一门科学。科学研究的帅才最重要的素质是洞察力（Vision和Insight），能够发现有前景的新领域或在新领域内发现和解决最关键的问题。下面仅举几例说明Dijkstra的洞察力。

　　解决[编译系统](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=9268696&ss_c=ssc.citiao.link)中的关键问题

　　FORTRAN是最早得到一定程度推广的高级语言，但FORTRAN的编译系统并未科学地解决一些主要的难题。最早提出用堆栈（stack）来编译复杂公式的是德国的Bauer和Samelson，他们的著名论文“顺序公式的翻译”（[Sequential](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=72887870&ss_c=ssc.citiao.link) Formula [Translation](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=101648341&ss_c=ssc.citiao.link)）是编译方面的经典论文。最近有些报道说Dijkstra是堆栈的发明人，这恐怕不符事实。Dijkstra发展了堆栈的概念，使之用于整个编译，以及[目标代码](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=55609876&ss_c=ssc.citiao.link)运行时的动态存储分配，并在此基础上和Jenson完成了世界上第一个ALGOL60编译系统，采用了他首创的[优先数](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=70014952&ss_c=ssc.citiao.link)编译算法。其中递归调用[子程序](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7539248&ss_c=ssc.citiao.link)时的环境维护是Dijkstra的重要贡献，Display这一术语就是当时他发明的，这是用来维护[动态环境](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=5192165&ss_c=ssc.citiao.link)的一组寄存器（软件），其结构清晰并能适应任何复杂情况。我于[20世纪60年代](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=371024&ss_c=ssc.citiao.link)初看完他的文章后马上想到，假如[计算机硬件](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=507169&ss_c=ssc.citiao.link)中有对应的设备将极大地提高软件的运行效率。

　　Dijkstra是ALGOL 60报告的主要起草者之一，1972年Dijkstra在他获得ACM[图灵奖](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=6042358&ss_c=ssc.citiao.link)的讲演中，仍对这一报告给予高度评价：“只有极少极少几个像ALGOL60报告这样短的文件能给计算机界带来如此深远的影响。”

　　同步进程的协调和操作系统的结构

　　Dijkstra在1950～1952年期间曾当过三年程序员，在从事[硬件中断](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=8095846&ss_c=ssc.citiao.link)处理程序的研制中，他发现一些程序错误在多个中断同时出现的情况下无法再现，很容易被当作硬件的瞬间故障，这一现象使Dijkstra毛骨悚然，促使他后来钻研用科学方法从事软件研制。操作系统是当时最复杂的软件，1965年Dijkstra在ACM通讯上发表了仅一页长的短文“并行程序的控制”，这是他在操作系统领域的第一个重要贡献。该文提出了并行程序互锁问题的一个解决方案。“死锁”（Deadly embrace）这一术语是Dijkstra发明的。1967年在首届[操作系统原理](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=55380790&ss_c=ssc.citiao.link)研讨会上，Dijkstra介绍了他和几个博士生研制的THE[多道程序系统](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7667605&ss_c=ssc.citiao.link)。THE系统的目的是验证Dijkstra关于操作系统原理、结构、同步[进程通信](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=60847&ss_c=ssc.citiao.link)机制等方面的一系列新想法。今天已经普遍采用的系统的多层结构、抽象、上层不需了解下层的详细细节等科学原则就是当时Dijkstra提出的，引起了强烈反响；同步进程通信的[信号量](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=398107&ss_c=ssc.citiao.link)[Semaphore](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=8217439&ss_c=ssc.citiao.link)这一术语也是Dijkstra当时创造的。

　　同步进程协调方面的程序错误是很难发现的，为此我曾吃过大苦头。1979年我开始设计Ⅱ型照排系统，用了两台双极型高速微处理器平行工作实现轮廓字形的复原和控制激光扫描，两台微处理器通过信号量彼此协调。设计和编程均无错误，但在把微程序翻译成[二进制代码](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=15555451&ss_c=ssc.citiao.link)时错了一位，结果造成了无法再现的随机故障：[激光打印机](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=102833&ss_c=ssc.citiao.link)输出的汉字有时拖尾巴。这一故障持续了一年多，引起用户不满，后由潍坊计算机公司的一位协作人员挖空心思地发现了错误的原因，即二进制码错误，问题才得以解决。

　　上世纪六十年代操作系统的研制经受了一系列重大挫折，典型的例子是OS 360。而Dijkstra为首的一批科学家的努力，使操作系统走上了健康发展的道路。

　　Go To语句有害和结构程序设计

　　结构程序设计被称为软件发展中的[第三个里程碑](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=156794149&ss_c=ssc.citiao.link)（第一、二个里程碑是子程序和高级语言）。早在1965年召开的IFIP会议上，Dijkstra就提出“Go To语句可以从高级语言中取消”，“一个程序的质量与程序中所含的Go To语句的数量成反比”。但是，Dijkstra讲话的影响很小，当时人们正忙于IBM 360系列的使用；而IBM 360的主要语言之一是FORTRAN，Go To语句则是FORTRAN的支柱。

　　1968年Dijkstra给ACM通讯写了一篇短文，该文后改成信件形式刊登，以便早日发表，这就是具有历史意义的、著名的“Go To Letter”。Dijkstra在信中建议：“Go To语句太容易把程序弄乱，应从一切高级语言中去掉；只用三种基本控制结构就可以写各种程序，而这样的程序可以由上而下阅读而不会返回”。这封信引起了激烈的讨论。人们逐渐认识到：不是一个简单地去掉Go To的问题，而是促进一种新的程序设计观念、方法和风格，以期显著提高软件生产率和降低软件维护代价。当时采用结构程序设计方法的两个最著名项目是：

　　1． 纽约时报信息库管理系统，含8.3万行源代码，只化了11人年，第一年使用过程中，只发生过一次使系统失效的软件故障；

　　2． [美国宇航局](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=449802&ss_c=ssc.citiao.link)[空间实验室](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=4774300&ss_c=ssc.citiao.link)操作的模拟系统，含40万行源代码，只用两年时间就全部完成。

　　上个世纪六十年代末到七十年代初，上述这两个系统可以算得上是大型软件了。

　　结构程序设计概念影响了后来的高级语言，也影响了一代程序员的风格和习惯。

　　坚持真理，不惜反对恩师

　　ALGOL 68语言是Van Wijingaarden为主设计的，得到了IFIP下属的工作小组的多数票通过。Wirth、Dijkstra和Hoare等工作小组成员发表了“[少数派报告](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=135726&ss_c=ssc.citiao.link)”，强烈反对这一语言及其表达方式。ALGOL 68的表达方式独特而严格,但不易理解，是Wijingaarden的发明。事实证明，少数派是正确的，ALGOL 68很快就夭折了。

　　要知道，Wijingaarden是Dijkstra的恩师。上世纪五十年代初Dijkstra在[阿姆斯特丹](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=64959781&ss_c=ssc.citiao.link)数学中心编程序时Wijingaarden是他的领导。Dijkstra当时正在攻读[理论物理学](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=477848&ss_c=ssc.citiao.link)位，他发现理论物理课程和程序设计在精力上越来越冲突，他又拿不定主意，到底成为一个理论物理学家还是选择程序设计作为自己的职业，而他当时又担心程序设计能否成为被尊敬的学科。充满不安心情的Dijkstra于1952年春的一个早晨敲开了Wijingaarden办公室的门。几个小时的谈话，使Dijkstra变了另外一个人，因为Wijingaarden向他指明了程序设计这门学科正处在起步阶段，因而有重大的机会，Dijkstra可以努力使程序设计变成一个受人尊敬的学科。Dijkstra把这次谈话称为他一生中的转折点，于是决定尽快结束理论物理研究，全身心地投入程序设计。Dijkstra说，为此他应该终生感谢Wijingaarden。但是Dijkstra还是强烈地反对其恩师的得意之作ALGOL 68，少数派报告虽被IFIP压制，但真理终究得到了证明，后来的成功语言都是背离ALGOL 68方向的。

　　Dijkstra的名言今天仍有现实意义

　　Dijkstra的主要贡献是在上个世纪五十年代末到七十年代初，也就是他二十多岁到四十岁出头这段时间完成的。回忆这些过去的成就是想强调：洞察力对一个科学家是何等的重要! Dijkstra获图灵奖以后，软件领域又涌现出[图形用户界面](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=64893102&ss_c=ssc.citiao.link)、[面向对象技术](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=592124&ss_c=ssc.citiao.link)等一系列新的里程碑，因特网更是带来一个全新的时代。但是三十年前Dijkstra关于程序可靠性的一些名言至今仍有意义：

　　“有效的程序员不应该浪费很多时间用于程序调试，他们应该一开始就不要把故障引入。”

　　“[程序测试](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=10656472&ss_c=ssc.citiao.link)是表明存在故障的非常有效的方法，但对于证明没有故障，调试是很无能为力的。”

　　Dijkstra大力鼓吹程序正确性证明，但这一方法离实用还有相当距离，因为一段[源程序](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7742252&ss_c=ssc.citiao.link)的正确性证明的文字往往比源代码还要长，所以充分的软件测试今天仍不可或缺。但是程序员的科学训练是十分重要的，有人曾做过一个试验：一个题目由一批印度程序员编程，其结果惊人地相似；而由一批中国程序员来做，编出的程序五花八门。中国的软件人员有时把创造性放在不恰当的地方。只有规范的科学的编程，一个大项目才能得到有效的管理，其质量才有保证。

问题三：列举计算机网络领域全世界最好的十个期刊，十个最好的会议，十个最牛的科学家或企业家，十个最好的学校，十个最好的企业。

期刊：《计算机仿真》，《计算机科学》，《自动化应用》，《计算机测量与控制》，《电脑知识与技术》，《Communications of the ACM 美国计算机学会通讯》，《IEEE transactions on image processing IEEE 图象处理汇刊》，《Pattern recognition 模式识别》，《IEEE ACM transactions on networking IEEE /ACM 网络汇刊》，《IEEE transactions on computers  IEEE 计算机汇刊》

会议：ACM Special Interest Group on Data Communication(SIGCOMM) ACM Special Interest Group on Mobility of Sys-tems, Users, Data and Computing (MOBICOM)

ACM Special Interest Group on Measurement and Evaluation (SIGMETRICS)  
ACM/IEEE Intl Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)  
ACM/IEEE World Wide Web Conf. (WWW)  
Knowledge Discovery and Data Mining (SIGKDD)  
IEEE Intl Conf. on Data Engineering (ICDE)  
Intl Conf. on Very Large DataBase (VLDB)  
Special Interest Group on Information Retrieval (SIGIR)  
ACM SIGMOD Conf. on Management of Data/Principles of DB Systems

科学家或企业家：金怡濂，姚期智，王小云，慈云贵，理查德·马修·斯托曼（Richard Matthew Stallman, RMS），林纳斯·本纳第克特·托瓦兹(Linus Benedict Torvalds)，冯·诺依曼（John Von Neumann），阿兰·麦席森·图灵（Alan Mathison Turing），克劳德·香农(Claude Elwood Shannon)，赫伯特•亚历山大•西蒙(Herbert Alexander Simon)

学校：麻省理工大学，斯坦福大学，卡耐基梅隆大学，剑桥大学，哈佛大学，加州大学伯克利分校，牛津大学，苏黎世联邦理工学院，新加坡国立大学，普林斯顿大学

企业：联想-IBM，惠普，贝尔，戴尔，惠普-康柏，英特尔，ACER宏碁，苹果，富士通，东芝

问题四：你对你的职业发展，专业期待等方面有何建议？（可选题）