论文题目	一个基础网络检测和分析工具的设计和实现

一、选题背景和意义

自冯·诺伊曼体系结构提出以来，计算机科学技术得到了长足发展，操作系统技术、数据库技术和计算机网络技术则是现今计算机科学领域中最重要的三块"基石"。其中，计算机网络技术利用通信设备和信道将地理位置不同的、功能独立的多个计算机系统连接起来，并在网络操作系统、网络管理软件和网络通信协议的管理和协调下，实现了网络中快速地信息传递和资源共享，极大地促进了信息时代的到来。如今，计算机网络已经普及到人们日常生活的方方面面，与电力、电话一样，成为维系现代社会正常运作的重要纽带之一，保证网络持续、高效和安全地运行变得至关重要。

网络故障诊断是网络管理的核心部分。由于通信协议、硬件、软件以及物理环境(如温度、湿度)等多种因素的影响下，网络系统可能会在许多不可预测的时刻发生故障。早期的故障诊断主要由专业技术人员人工进行 —— 首先，技术人员对系统进行不间断的观察，来获取各种网络资源和网络节点的状态信息；然后，对获取的信息进行筛选过滤；最后，结合过往的经验和直觉得出故障诊断结果。人工诊断的方式往往代价高昂且诊断效率低下。随着网络规模的扩大，网络逐步向异构性、动态性和智能化的方向发展,这些变化对网络故障诊断提出了越来越高的要求，这种方式已经很难满足高强度、快节奏的现代信息社会的需求。因此迫切需要一个可以在网络出现故障时指导管理人员以最小代价、最快速度定位故障、排除故障的智能辅助决策系统，并且该系统可以通过知识的积累，不断提高系统的故障诊断能力，降低对网络管理人员的要求。通过提高网络故障诊断能力，实现快速准确的故障诊断，对提高网络系统的鲁棒性和系统服务的可用性至关重要，对减少网络故障的修复时间和降低网络的维护成本也具有重要意义。

实现网络快速故障诊断的主要障碍之一就是复杂网络故障诊断中广泛存在的不确定性问题。要使智能故障诊断技术走向工程实际，首先必须解决不确定性问题的求解。目前，国内外学者对故障诊断技术，进行了深入的研究，提出了很多行之有效的理论和方法。从方法上看一般采用序贯搜索技术，就优化准则来说主要有基于信息量、基于风险、基于时间开销、基于费用开销等准则，就推理模型来说主要有故障树、决策树、马尔柯夫可靠性模型、信息模型、故障传播有向图模型、Petri 网、概率论因果网络、节约覆盖集等。所涉及的理论主要包括概率理论、模糊理论、信息理论、图论、多目标优化决策理论、序贯决策论。所有方法都遵守"最小代价"原则，具体目标一致，即期望的故障发现率和定位率最高，所有诊断时间最短，搜索成本最低。

在企业实践中发现：用户通过网络访问某共享资源出现问题时，往往会首先将问题归结于网络问题。而如何在最短的时间内，明确问题是网络问题、诊断问题原因以及提出解决或缓解问题的方案，常常使网络管理员焦头烂额。本课题将致力于开发一个在问题发生时可以指导管理人员以最小代价、最快速度明确问题、定位原因和解决问题的网络检测工具。

二、课题关键问题及难点：

根据在企业实践的经验，构建完善的网络检测工具主要存在三个方面的难点:

1. 信息获取。

(1)在网络问题检测过程中，信息获取是一个重要环节。网络系统一般由很多不同的组件构成，如主机、路由器、交换机等，组成结构错综复杂，不同组件之间相互关联耦合，存在大量的不确定因素，包括通信协议、硬件、软件和物理环境等。因此，对网络系统进行观测往往能够获取大量的信息，但是其中不乏冗余、无效的数据需要过滤。

(2)对于不同的硬件和操作系统，获取网络数据的接口支持可能是各异的，如何实现跨平台、多源的信息获取也将是一个巨大的挑战。

综上所述，受到各种因素的限制，能获得的有效观测数据是有限的，而且往往可能是不完整的、不精确的。

2. 信息处理。

简单地说，诊断就是由问题现象获得最终问题原因的推理过程，问题现象和问题原因之间往往具有一定的因果关联关系。然而，对于复杂网络而言，这种因果关联关系并不一定是确定的映射关系，而往往由于诊断对象的复杂性、测试手段的局限性、知识表达的不精确，使得问题现象和问题原因之间的映射表现为随机性和不确定性。同时，信息获取也存在诸多的限制，因此，如何在不充分、不完整、不确定的信息条件下完成决策模型的推理是信息处理面临的关键问题。

3. 仿真网络。

本课题构思来源于实践中，但是由于时效性的要求，在问题发生时，往往首先专注于探索问题的解决或缓解方案，对"现场事件"数据未曾进行系统地记录和存储。因此，在构建网络检测工具的网络诊断模块时，缺乏大量的先验数据来构建和训练用于处理数据的数学模型。网络检测工具开发完成后，必须测试和验证工具的有效性和准确性，避免对正常的网络环境产生意料之外的影响。

因此，必须构建一个仿真的网络环境，通过注入各种故障来获取先验数据，并作为测试环境对网络检测工具进行验证。

注：开题报告可单独装订，但在院（系）范围内，封面和装订格式必须统一。

三、文献综述（或调研报告）：

计算机网络检测技术研究综述 ：

随着网络规模的扩大, 网络逐步向异构性、动态性和智能化的方向发展, 国内外学者对网络检测技术也进行了很多深入的研究化。根据各种研究基于的网络场景, 待检测的网络对象可以归纳为规模依次增大的家庭网络[7]、企业网络[5]和数据中心网络[6][8][14][18], 保持各个网络实时畅通是提高用户体验的重要前提。正常情况下, 提供给用户的网络应用和服务运行良好, 但有时会出现一些异常行为, 通常表现为用户访问延迟增加、访问无应答等现象[5]。这些异常行为往往是底层网络中通信出现瓶颈的表现, 但是也无法排除应用程序本身漏洞、系统资源不足等其他问题的可能性。网络检测的目标就是发现这些异常行为, 并区分和定位产生它们的根本原因。基于计算机系统的性能指标以及资源的利用率, 对异常行为进行图表化, 根据离散点或连续点的图像特征划分了 Point, Collective, Contextual 和 Pattern 四种异常类型[12]。还描述了限制系统性能的资源或应用, 并将之称为瓶颈, 包括资源饱和瓶颈以及资源争用瓶颈[12]。造成这些异常和瓶颈的原因分为以下几类 ：

(1)应用错误 —— 例如对配置文件进行错误的更改, 软件更新或代码存在错误等应用层的问题[5], 可能会触发意料之外的资源瓶颈事故。

(2)工作负载 —— 指应用由于突发原因在某一时刻显著偏离平均或预期的工作负载强度, 可能会导致网络流量的拥塞控制出错, 以及线程资源的过量分配等问题[6]。

(3)平台架构 —— 指短时间内在底层系统架构或操作系统发生的瞬态事件。例如, 存储器硬件错误, JVM"垃圾回收"引发资源瓶颈等。

(4)系统错误 —— 一般可归因于系统软件错误, 操作员错误, 硬件故障, 环境问题和安全违规[14]。这类故障可能是间歇性的, 突发的或永久的, 并可能会严重影响应用性能。 网络检测技术中包含两个关键内容, 分别是数据采集和数据分析。数据采集的数量和质量往往会直接影响到数据分析的有效性和准确性。

数据采集是对系统进行系统性、规律性地监控和采样, 从而获得一组时间序列上与系统相关的性能指标值。性能指标是系统的或者某个能够表示系统状态的组件的一个属性, 往往可以用一组与系统密切相关的性能指标值向量来描绘系统特定时刻发生的一个问题[12]。基于数据的粒度大小, 将大多数研究中采集的数据分为两个层次: 应用性能指标[5][17]和系统底层性能指标[1-5]。应用程序性能指标是用来描述当前应用程序或进程的运行状态, 例如, 响应时间, 吞吐量; 系统底层性能指标是指表示当前底层硬件或操作系统状态的属性值, 例如, CPU 使用率, IO 等待时间等。

数据采集通过使用操作系统内核内建性能计数器(例如)或安装自动化第三方客户端工具, 收集应用程序和操作系统的性能数据。采集方式也分主动和被动[12][20]。主动是指主动向网络中注入测试流量, 直接测量需要的性能指标, 而无需等待特定事件发生。例如，Pingmesh[14]在服务器上部署 Pingmesh Agent 工具定期发起基于 TCP 协议的 Ping 获取网络延迟和丢包个数。这种方式需要最小化注入流量对正常网络通信的影响, 其影响程度往往取决于流量注入的大小和频率。另外测试流量需要尽可能类似于实际流量, 否则, 采集到的性能数据可能与网络实际运行状态不完全匹配。被动则是依赖网络设备(例如, 路由器和交换机)或终端主机观测实际的系统状态或网络行为, 可能需要等待很长时间才能观测到特定的事件数据。例如, [3]和[6]中收集网络事件日志(如: syslog)作为数据来源。鉴于主动和被动的特性,一般同时使用主动和被动的方式采集数据。

在实际网络中, 很多性能指标的可观测性很大程度上取决于基础网络结构的组成和应用服务提供商的服务配置。例如, 在数据中心网络中可以直接观测网络底层基础设施甚至应用服务源代码, 而在云环境中, 应用服务对于云提供商是一个"黑盒子", 应用服务提供商则缺少

虚拟机之外的全局网络架构视图。

结合现有文献[1][11][12], 对过往检测系统采用的数据分析策略进行了总结 :

(1)基于模型的检测(Model-Based Detection)。基于模型的推理中, 检测分析建立在面向对象的模型基础之上, 目标系统的实体都将模型化为诊断对象, 这些诊断对象之间的关联关系被清晰地描述, 从模型上也反映了实际系统的结构行为。基于模型的故障检测问题来源于模型和实际系统的差异, 模型是对现实系统物理结构及其构成的理想描述, 构成模型的基本成分要满足一定的规则。在检测的过程中, 首先假设物理系统的所有组成部分都遵循模型的约定, 通过度量和观察结构发现实际系统和理想模型之间的差异, 最后假设其中的某些组成部分没有满足模型的约定, 从而达到能够解释实际系统和理想模型之间偏差的目的。

(2)基于案例的检测[15](Cased-Based Detection)。基于案例的检测是通过过去求解类似问题的经验和知识获得当前问题结果的一种检测模式。基于案例的检测系统通常由案例索引机制、检索机制、案例改写和案例库四个核心功能部件构成。其中, 案例库提供支持问题求解的一组案例, 它是系统过去进行问题求解经验的聚集。根据问题描述, 案例检索机制从案例库中查找一个与当前问题相匹配的案例, 如果该案例满足问题描述的要求, 则输出相应的结果, 否则根据问题描述, 对检索出的案例进行修改, 案例改写的结果形成一个满足全部问题描述要求的答案，该结果同时作为一个新的案例经索引机制组织到案例库中以备将来使用。

(3)基于编码的检测(Code-Based Detection)。每个故障的产生都会引起大量症状事件的发生, 每个对象产生的故障症状事件可能是该对象自身故障问题引发的本地事件, 也可能是有关联关系的对象出现故障, 传播过来引发的事件。处理这些由故障而引发症状事件的方法是把事件看成一个标识故障的"密码", 相关性分析的过程就是对症状事件进行解码的过程, 最终确定密码标识的、出现症状事件的故障列表。

(4)依赖关系图模型[5][20](Dependency Graph Model)。在网络检测系统中, 故障往往不是显而易见的。系统常常监测到一些故障表现(异常), 但很难根据这些异常来直接定位故障源, 这些都是由于网络中被管对象之间的相互依赖关系导致故障传播的结果。依赖关系图可以表示为有向图 G=(O,D), 其中 O 是有限非空的对象集合, D 是对象之间边的集合, 若有直连边(o_i, o_j)属于 D, 表示对象o_i上发生的错误或故障将导致对象o_j上故障的发生, 每个有向边标注的条件概率表示有向边的头节点和尾节点之间的依赖程度。依赖关系图为故障检测提供了一般性的自然描述, 对网络系统的模拟简明了很多。

计算机网络检测系统是计算机网络管理系统的核心模块之一, 至今已有 20 余年的研究历史。本文首先归纳了造成互联网用户体验下降的常见问题原因, 然后概括性地描述了网络检测技术中数据采集的类型、方式和挑战, 最后对过往网络检测系统采用的检测策略进行了总结。本课题将致力于开发一个在问题发生时可以指导管理人员以最小代价、最快速度明确问题、定位原因和解决问题的网络检测工具。

参考文献 :

[1] 陈琳. (2005). 网络故障诊断关键技术的研究. J]. 国防科学技术大学.

[2] 郑秋华. (2007). 网络故障智能诊断关键技术研究 (Doctoral dissertation).

[3] Yamanishi, K., & Maruyama, Y. (2005, August). Dynamic syslog mining for network failure monitoring. In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (pp. 499-508). ACM.

[4] Sommers, J., Barford, P., Duffield, N., & Ron, A. (2005, August). Improving accuracy in end-to-end packet loss measurement. In *ACM SIGCOMM Computer Communication Review* (Vol. 35, No. 4, pp. 157-168). ACM.

[5] Kandula, S., Mahajan, R., Verkaik, P., Agarwal, S., Padhye, J., & Bahl, P. (2009). Detailed

diagnosis in enterprise networks. ACM SIGCOMM Computer Communication Review, 39(4), 243-254.

[6] Gill, P., Jain, N., & Nagappan, N. (2011). Understanding network failures in data centers: measurement, analysis, and implications. ACM SIGCOMM Computer Communication Review, 41(4), 350-361.

[7] Calvert, K. L., Edwards, W. K., Feamster, N., Grinter, R. E., Deng, Y., & Zhou, X. (2011). Instrumenting home networks. ACM SIGCOMM Computer Communication Review, 41(1), 84-89.

[8] Wu, X., Turner, D., Chen, C. C., Maltz, D. A., Yang, X., Yuan, L., & Zhang, M. (2012, August). NetPilot: automating datacenter network failure mitigation. In Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication (pp. 419-430). ACM.

[9] Kim, H., & Feamster, N. (2013). Improving network management with software defined networking. IEEE Communications Magazine, 51(2), 114-119.

[10] Mysore, R. N., Mahajan, R., Vahdat, A., & Varghese, G. (2014). Gestalt: Fast, Unified Fault Localization for Networked Systems. In 2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14) (pp. 255-267).

[11] Lee, S., Levanti, K., & Kim, H. S. (2014). Network monitoring: Present and future. Computer Networks, 65, 84-98.

[12] Ibidunmoye, O., Hernández-Rodriguez, F., & Elmroth, E. (2015). Performance anomaly detection and bottleneck identification. ACM Computing Surveys (CSUR), 48(1), 4.

[13] Bajpai, V., & Schönwälder, J. (2015). A survey on internet performance measurement platforms and related standardization efforts. IEEE Communications Surveys & Tutorials, 17(3), 1313-1341.

[14] Guo, C., Yuan, L., Xiang, D., Dang, Y., Huang, R., Maltz, D., ... & Lin, Z. W. (2015, August). Pingmesh: A large-scale system for data center network latency measurement and analysis. In ACM SIGCOMM Computer Communication Review (Vol. 45, No. 4, pp. 139-152). ACM.

[15] Kanuparthy, P., Lee, D. H., Matthews, W., Dovrolis, C., & Zarifzadeh, S. (2013). Pythia: detection, localization, and diagnosis of performance problems. IEEE Communications Magazine, 51(11), 55-62.

[16] Chen, A., Wu, Y., Haeberlen, A., Zhou, W., & Loo, B. T. (2016, August). The good, the bad, and the differences: Better network diagnostics with differential provenance. In Proceedings of the 2016 ACM SIGCOMM Conference (pp. 115-128). ACM.

[17] Arzani, B., Ciraci, S., Chamon, L., Zhu, Y., Liu, H. H., Padhye, J., ... & Outhred, G. (2018). 007: Democratically finding the cause of packet drops. In 15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18) (pp. 419-435).

[18] Kimura, T., Watanabe, A., Toyono, T., & Ishibashi, K. (2018). Proactive failure detection learning generation patterns of large-scale network logs. IEICE Transactions on Communications.

[19] Yu, M. (2019). Network telemetry: towards a top-down approach. ACM SIGCOMM Computer Communication Review, 49(1), 11-17.

[20] Salah, S., Maciá-Fernández, G., & Díaz-Verdejo, J. E. (2019). Fusing information from tickets and alerts to improve the incident resolution process. Information Fusion, 45, 38-52.

四、方案（设计方案、或研究方案、研制方案）论证：

本课题设计并实现一个基于网络用户视角的网络检测系统 —— Woodpecker。

1. 应用场景

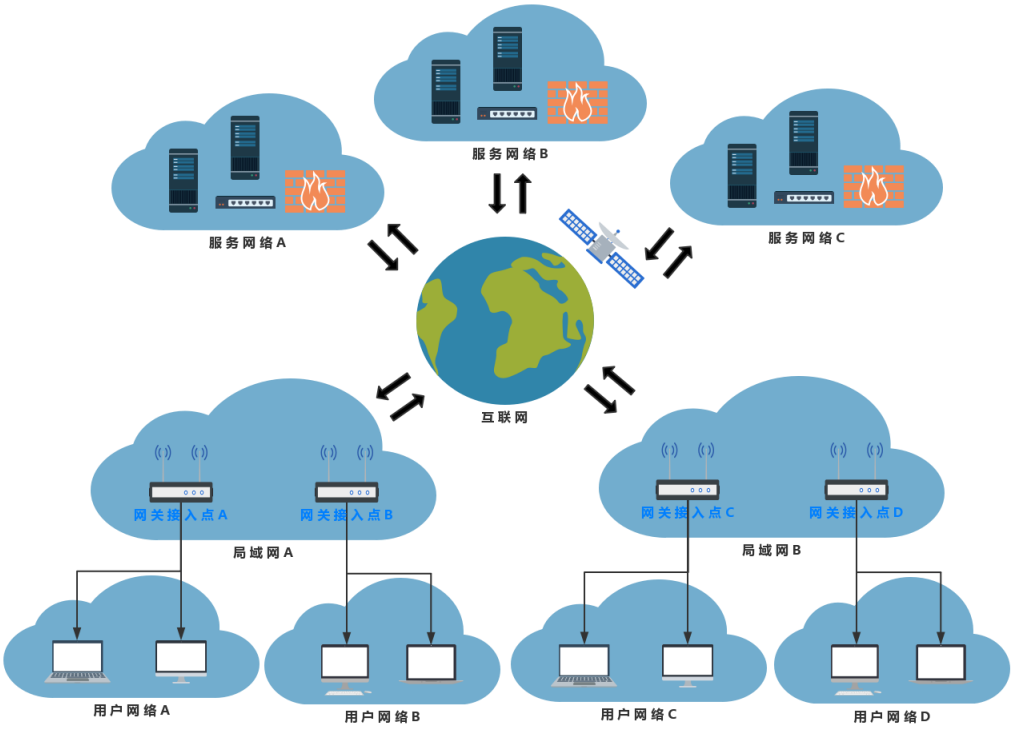

图 1. 基于网络用户视角的抽象通信网络

如图 1 所示，基于网络用户视角:

(1) 终端设备经由网关接入互联网,一个网关一般支持同时接入多个终端设备。

(2) 多个终端设备只有接入相同的网关,才认为它们处于相同的网络环境,称为用户网络。

(3) 抽象地,根据网关将端到端的通信网络划分成三部分: 用户网络、互联网和服务网。

(4) 互联网是透明的，包括网关所在的局域网(例如校园网络，企业网络等)。

(5) 服务网络是对提供特定网络服务集合的服务器集群及其网络设施的整体抽象。

Woodpecker 基于网络用户视角，直接识别造成网络用户体验下降的异常(如网页加载失败、视频加载缓慢)，并通过分析异常发生前后采集的数据，分析和定位导致异常发生的故障源位置和具体原因 —— 用户网络、互联网或者服务网络。

2. 系统架构

如图 2 所示, Woodpecker 的系统架构由五个模块组成 ：

(1) 数据采集模块

数据采集模块在用户网络的终端设备上部署客户端(Monitor)，来收集实时的相关数据。一方面, Monitor 监控终端设备的进出流量和性能指标，分别按照应用进程和服务网络进行关联、聚集、统计和分析。另一方面, Monitor 采取主动轮询的策略，对终端设备近期常访问的服务网络周期性地发起探测报文，收集延迟、丢包和路由等数据。

在条件允许的情况下，数据采集模块的监控范围可扩展至用户网络所属的局域网内，利用 SNMP、sFlow、NetFlow 等协议收集局域网内基础网络设施的相关数据，构成分层数据采集模型。

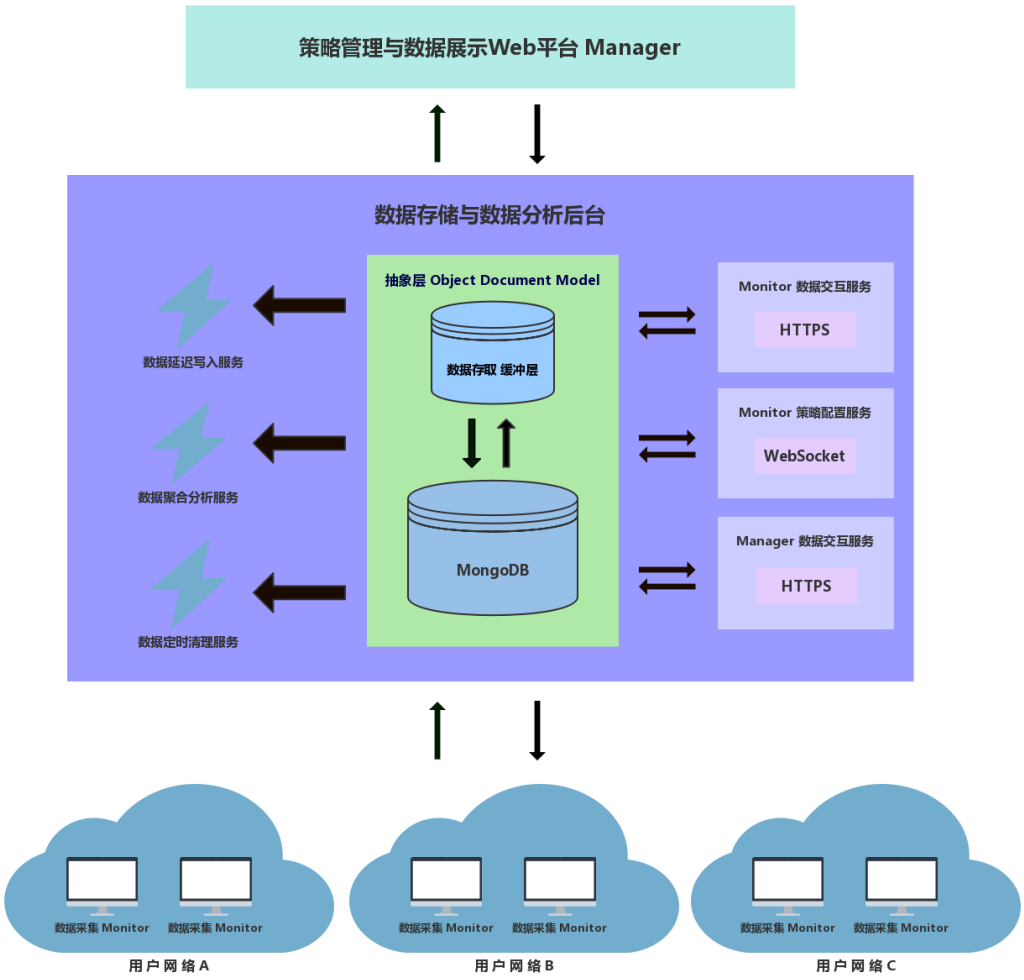

图 2. Woodpecker 运行架构

(2) 数据存储模块

Woodpecker 后台采用 MongoDB 存储客户端采集的数据，并选用 Redis 或 Apache Kafaka 作为数据存储缓冲层。其中，MongoDB 是非关系型数据库，数据结构直接采用 JSON 格式，一方面，降低了数据库设计的复杂度；另一方面，实验显示，MongoDB 具有良好的查询性能。引入数据存储缓冲层，则是借助内存型数据库良好的写入性能，降低 Monitor 提交数据时的响应时间。

数据存储模块对上述存储策略进行封装，提供数据查询和写入的接口，并实现定时清理过期数据的守护进程。

(3) 数据展示模块

Woodpecker 基于 Vue.js 开发 Web 界面，包括数据展示界面和策略管理界面。基于数据查询接口，数据展示模块首先对数据进行简单地处理，然后用 Echarts.js 模板进行可视化渲染，最后交由前端 Web 界面展示。数据展示界面的可视化粒度分为终端设备、用户网络和局域网，聚合范围和抽象层次依次增大。

(4) 策略管理模块

策略管理模块是 Woodpecker 为管理员提供的系统配置界面。Woodpecker 的可配置项主要包括数据采集模块的运行策略，数据存储模块的存储策略以及数据展示模块的展示策略。通过策略调整，一方面，控制 woodpecker 客户端对实际网络可能造成的工作负载；另一方面，来应对 woodpecker 后台可能出现的各种突发状况。此外，策略管理模块还负责进行多用户访问权限管理。

(5) 数据交互模块

数据交互模块提供进行数据通信的接口，主要用于客户端上传数据，后台数据查询以及策略调整下发。其中，上传数据接口和数据查询接口采用的是 RESTful API，鉴于对安全性的要求，采用 https 协议进行通信。策略调整下发则要求建立一个双向通信连接，即允许服务端主动向客户端推送数据，因此，采用 WebSocket 协议来实现。

3. 开发阶段

根据预期的 woodpecker 完成程度，将开发过程划分为以下五个阶段 ：

1) 定义能够充分描述计算机网络通信特征的指标集合。
2) 精确的、低耗的、全面的和多平台的数据采集工具。
3) 实时的、简洁的、清晰的和多用户的数据展示平台。
4) 明确现象、故障、原因之间的层级划分及相互关系。
5) 充分的数据，完善的算法，实时且准确的故障定位。

五、进度安排：

起止日期	工作内容
2019.02.18-2019.02.25	完成论文翻译工作
2019.02.26-2019.03.15	搜集并阅读文献资料，进行前期调研
2019.03.15-2019.03.26	完成开题报告
2019.04.09-2019.04.15	填写中期检查表
2019.03.27-2019.04.05	完成数据采集客户端
2019.04.06-2019.04.15	完成数据存储与分析后台
2019.04.15-2019.04.25	完成策略管理与数据展示平台
2019.04.05-2019.04.30	搭建测试环境，并完成系统测试
2019.04.15-2019.05.10	撰写完成论文
2019.05.11-2019.05.15	修改论文，完成终稿
2019.05.16-2019.06.06	完成毕业答辩