FogBus：基于边缘和雾计算的基于区块链的轻量级框架

摘要：最近，人们更加重视集成Edge，Fog和Cloud基础架构，以支持各种延迟敏感和计算密集型的物联网（IoT）应用程序的执行。尽管不同的现实世界框架试图协助这种集成，但它们在平台独立性，安全性，资源管理和多应用程序执行方面存在局限性。为了解决这些限制，我们提出了一个名为FogBus的框架，该框架可促进端到端IoT Fog（Edge）云集成。FogBus为IoT应用程序和计算实例提供了平台无关的接口，用于执行和交互。它不仅帮助开发人员构建应用程序，而且还帮助用户一次运​​行多个应用程序，并帮助服务提供商管理其资源。此外，FogBus应用了区块链，身份验证和加密技术来保护敏感数据的操作。由于其简化的跨平台软件系统，因此易于部署，可扩展且具有成本效益。我们通过创建一个计算环境来演示FogBus的有效性，该环境将手指脉搏血氧仪作为物联网设备与基于智能手机的网关和基于Raspberry Pi的Fog节点进行集成，以进行睡眠呼吸暂停分析。我们还将通过部署实际的IoT应用程序，评估与其他现有框架有关的FogBus特性以及各种FogBus设置对系统参数的影响。实验结果表明，FogBus相对较轻且响应迅速，并且不同的FogBus设置可以根据情况要求调整计算环境。

1. 引言

物联网（IoT）范例使不同的传感器，机器和对象能够感知外部环境并通过Internet交换数据。它使用应用程序支持数据集成和分析，以识别事件并通过执行器触发必要的动作。因此，物联网可以帮助人类在不同的应用领域（如医疗保健，运输，公用事业基础设施，农业和监视管理服务）中进行有限的干预来构建网络物理系统（Gubbi等，2013）。它导致实现更智能的环境，例如智能健康，智能城市，智能交通，智能电网，智能农业和智能安全，从而实现实时决策。例如，用于灾难检测和预防的监视需要在严格的期限内完成IoT数据分析，以使响应团队能够快速及时地采取行动。根据目前的趋势，预计到2025年，此类智能环境将整合超过1万亿个IoT设备，对延迟敏感型应用程序的需求将增加50％（McKinsey＆Company，2018年5月）。

通过为不同的智能系统托管应用程序，现有的云数据中心可以提供一种处理大量地理上分散的物联网设备的方法（Muhammad等，2017）。然而，云数据中心与物联网设备之间的距离为多跳，这增加了数据传输和应用服务交付中的通信延迟（Afrin等人，2015）。对于诸如远程病人监视，森林火灾检测和交通信号管理等对时延敏感的应用，这种高时延通信会严重影响服务质量。此外，物联网设备可以在最短的时间内生成大量数据。当大量IoT设备通过Internet同时将数据传输到Cloud数据中心时，会发生严重的网络拥塞。为了克服以云为中心的物联网模型的这些局限性，出现了雾和边缘计算范式（Bonomi等，2012）。两者都利用边缘计算资源来实时决策和执行IoT应用程序。具有计算处理器的智能设备（例如Raspberry Pi设备，个人计算机，移动电话，路由器和微数据中心）可以提供潜在的边缘资源（Mahmud等，2018b）。基于在边缘资源上运行的操作，一些人认为Fog和Edge计算范式是相似的并且可以互换使用，而其他人则将Edge计算视为Fog计算的子集（Chang等，2019），这与观点一致由我们在本文中采用。

雾环境管理物联网设备和云数据中心之间的中间层，并支持表现出不同特征的各种物联网应用。其中一些应用程序需要大量计算，而另一些则需要大量网络或存储。Fog的计算实例称为Fog节点，并以分布式方式跨边缘网络部署。Fog节点中的物理资源，不同的面向服务的框架和各种应用程序可帮助Fog扩展类似于云的服务模型，例如基础设施即服务（IaaS），平台即服务（PaaS）和软件即服务（SaaS）物联网数据源（Yi等，2015b）。因此，它减少了服务延迟和网络拥塞，并改善了服务质量（QoS）和用户体验（Mahmud等人，2018c）。但是，大多数Fog节点在其计算能力，操作系统和对等通信标准方面受到资源限制和异构（Gupta et al。，2017）。在资源有限的情况下，不可能在Fog层容纳每个IoT应用程序。因此，需要将支持物联网的系统与Fog和Cloud基础设施无缝集成，以便可以根据应用程序的需求来利用边缘和远程资源（Mahmud等人，2018a）。在这种集成中，当以更高的频率接收物联网数据进行处理时，以云为中心的自上而下的方法来管理基于雾的资源变得不可行。在这种情况下，遵循分布式自下而上的方法，而不是依靠集中的资源管理策略，而是在本地进行决策并配置资源是有效的。此外，在集成环境上放置和执行应用程序时，内部和外部操作都会因计算实例的异构性而受阻。在这种情况下，独立于平台的技术可以克服点对点通信和应用程序运行时环境的障碍。但是，要实现超越基础架构和平台级别多样性的集成环境以及分散的资源管理策略，是一项艰巨的任务。由于多个决策实体的共存，不了解网络拓扑安全性和可伸缩性问题，其复杂性进一步加剧。

在文献中，存在许多实现软件框架的工作，这些软件框架用于集成支持IoT的系统，雾和云基础设施（Rahmani等人，2018; Dubey等人，2017; Yangui等人，2016; Bruneo等人， 2016）。但是，这些框架几乎不支持同时执行多个应用程序和平台独立性。而且，它们为应用程序开发人员和用户提供了狭窄的范围，可以根据个人需求调整框架。这些框架利用云资源进行数据存储，并经常迫使能源受限的物联网设备处理原始数据。为了减少管理开销，现有框架应用了集中化技术，这些技术最终会降低QoS。它们还将注意力集中在几个安全方面，从而增加了集成环境的脆弱性。为了克服可用框架的此类限制，我们开发了一个名为FogBus的轻量级框架。FogBus通过利用各种边缘网络资源，可以端到端实施集成的IoT- Fog-Cloud环境。它支持平台独立的应用程序执行和节点到节点的交互。它旨在帮助实现（a）不同的并发/应用程序编程模型，例如SPMD（单程序和多数据），工作流和流，以及（b）资源管理和调度策略，以在Fog中执行这些类型的应用程序和云集成计算环境。为了确保数据的完整性，保护和隐私，FogBus还实施了区块链（Zyskind等，2015），并采用了身份验证和加密技术，从而提高了其可靠性。

这项工作的主要贡献如下：

•我们提出了一个名为FogBus的轻量级框架，用于集成支持IoT的系统，Fog和Cloud基础架构，以根据应用程序需求利用边缘和远程资源。它应用了区块链来确保传输机密数据时的数据完整性。

•我们设计了独立于平台的应用程序执行和节点到节点的交互架构，以克服集成环境中的异构性。

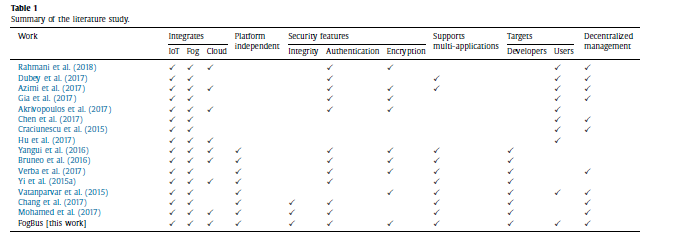
•我们开发了用于睡眠呼吸暂停数据分析的原型应用程序系统，评估了FogBus的特性，并说明了如何通过使用FogBus的不同设置在集成计算环境中处理由SPMD模型组成的应用程序（在医疗领域）如何实现。

本文的其余部分安排如下。第2节重点介绍了现有框架的关键要素，并将其与建议的框架进行了比较。在第3节中，提供了FogBus框架的描述。第4节将讨论FogBus的设计和实现。在第5节中，介绍了睡眠呼吸暂停分析的案例研究。第6节和第7节分别从框架特征和应用程序部署方面评估FogBus。第8节总结了提出改进FogBus的未来工作的论文。

1. 相关工作

将不同的支持物联网的系统与雾和云基础设施集成在一起的现有框架大致分为两类。第一种专注于特定于应用程序的原型，而另一种则提供了通用的PaaS模型。表1简要介绍了这些框架。Rahmani等。（2018）为基于IoT的医疗保健系统开发了基于原型的框架，该框架具有智能网关的架构，以促进本地存储和数据处理。在此框架中，云充当了数据分析和决策的后端基础架构。该框架取决于操作系统级别的安全功能。Dubey等人开发了另一个智能医疗保健的原型框架。（2017）。框架中使用了Intel Edison和Raspberry Pi设备作为Fog节点。通过基于角色的身份验证，可确保数据隐私。在此框架中，部分采用了Cloud来存储数据。Azimi等。（2017）还讨论了卫生保健解决方案的分层原型框架。健康分析分为两部分，分别放在Cloud和Fog基础架构中。该框架遵循MAPE-K模型进行计算，以支持各种应用程序的执行并提供数据加密。

此外，Gia等。（2017）提出了一种低成本的健康监测框架，以促进物联网数据的自主分析。在该框架中，物联网设备具有计算能力，并且可以预处理原始数据。他们还可以将数据转发到能耗更低的不同Fog节点。在Fog层中，管理分布式数据库的数据分类和安全性。Akrivopoulos等。（2017）开发了另一个原型框架，该框架允许用户共享健康数据并在紧急情况下进行通知。框架中的每个操作都由驻留在云中的Spark IoT Platform Core管理。该框架使用加密和身份验证技术来提高安全性。



Chen等。（2017）和Craciunescu等人。（2015年）分别开发了分别用于智能城市监控和气体泄漏监控系统的原型框架。在这两个框架中，雾基础设施都进行必要的数据处理和决策操作。这些框架中部分采用了远程云来管理IoT数据记录。但是，作者没有提到任何用于管理框架内Fog节点的异构性的安全功能或技术。同样，胡等。（2017）省略了他们开发的人脸识别系统原型框架的安全问题。在此框架中，集中式云管理集成环境的所有资源，并将部分计算任务卸载到Fog基础架构。在Fog上完成任务后，仅将结果发送回Cloud进行进一步分析和存储。为了促进IoT，Fog和Cloud的集成，Yangui等人开发了以云为中心的PaaS框架。（2016）自动化应用程序的配置。PaaS有助于开发各种应用程序，部署和管理Fog节点。该框架可以处理节点的异构性。此外，该框架还扩展了Cloud Foundry架构的安全功能。同样，Bruneo等。（2016年）提出了一种以雾为中心的PaaS框架Stack4Things，用于在计算良好的IoT设备上部署和执行多个应用程序。在那里，雾基础设施充当了集中式可编程协调器。该框架应用了基于云的安全功能，并处理了超过实例异构性的各种应用程序。此外，Verba等。（2017）提出了一种网关架构，该架构提供了用于集成Fog节点和IoT设备的PaaS。网关通过身份验证协助消息传递通信。该框架支持网关和Cloud数据中心的水平集成，以进行应用程序部署和任务迁移。Yi等。（2015a）还提出了一个用于集成环境的基于Cloudlet的全面PaaS框架。该框架要求运行资源丰富的Fog节点，并且要求对每个节点（包括IoT设备）进行虚拟化。现有的身份验证技术有助于确保框架操作的安全。另外，Vatanparvar等。（2015年）开发了一种PaaS框架，该框架可管理Fog基础设施的家庭和微电网级别的用电量。它处理Fog节点和IoT设备的异构性，并加密数据。

由Chang等人提出的名为Indie Fog的PaaS框架。（2017）利用用户的网络设备执行IoT应用程序。在此框架中，基于应用程序需求，核心服务和资源管理指令已从Cloud数据中心扩展到Fog节点。它支持Fog雾节点，并结合了用户的手持设备。为了安全起见，它运行注册表服务。Mohamed等。（2017）还讨论了一种面向服务的框架，该框架通过Fog和Cloud基础设施管理基于智能城市的服务。在此框架中，服务分为两种类型。第一个管理框架的核心操作，包括资源管理和安全性。另一种类型则包含特定应用程序的要求。框架的安全性通过身份验证和访问控制机制来确保。在上述框架中，安全问题是从狭perspective的角度出发的，边缘和远程资源的计算能力都没有得到充分利用。由于将计算推向物联网设备或资源丰富的Fog节点，它们的总体部署成本和能耗也会增加。此外，大多数框架都忽略了计算基础架构中的异构性，因此它们很难同时支持多个应用程序。但是，我们提出的FogBus框架为用户和提供者都提供了服务调整工具。它可通过区块链确保数据完整性，并协助用户身份验证和数据加密。FogBus扩展了适用于不同应用程序的执行平台，从资源受限的Fog节点到云数据中心，超越了它们的多样性。

1. Fogbus框架

FogBus框架通过软件组件集成了各种硬件工具，这些组件提供了结构化的通信和独立于应用程序的平台执行。使用FogBus的集成IoT-Fog-Cloud环境的高层视图如图1所示。它包括以下元素。

**3.1硬件仪器**

下文讨论了构成FogBus基础的物联网设备，雾网关节点（FGN），雾计算节点（FCN）和云数据中心等硬件工具。

*物联网设备*：物联网设备包含感知外部环境的传感器和将任何给定命令转换为物理动作的执行器。通常，物联网设备受到能源和资源的限制，只能充当数据生产者或消费者。在某些情况下，IoT设备具备有限的计算能力来预处理原始数据。FogBus允许IoT设备通过Zigbee，Bluetooth和NFC等无线或有线通信协议与附近的网关连接。IoT设备的感测频率可以根据系统的上下文进行调整，其中IoT数据的格式随设备而异。

*雾网关节点（FGN）*：在FogBus中，雾网关节点（FGN）是分布式计算基础结构的入口点。FGN帮助IoT设备配置集成环境，以放置和执行相应的应用程序。通过FGN，FogBus提供了应用程序的前端程序，以便用户可以设置身份验证凭据，访问后端程序，传达服务期望，接收服务结果，管理IoT设备并根据其可承受性请求资源。此外，FGN进行数据过滤并以通用格式组织它们。FGN还汇总了从智能系统的不同来源接收到的数据。对于大规模处理，FGN将数据转发到集成环境的其他计算实例。FGN通过约束应用协议（CoAP）或简单网络管理协议（SNMP）与可访问的Fog节点保持快速，动态的通信（Slabicki和Grochla，2016年）。

*雾计算节点（FCN）*：FogBus可以同时处理许多异构的雾计算节点（FCN）。FCN配备了处理核心，内存，存储和带宽，可以执行各种FogBus操作。基于这些操作，FCN扮演三种不同的角色：

1.代理节点：为了运行IoT应用程序的后端程序，FogBus促进了相应的FGN与任何可访问的FCN连接。如果该应用程序中所需的资源可用，则此FCN会启动数据处理。如果它不满足应用程序的要求，则作为代理节点，它会与其他FCN和Cloud数据中心进行通信，以提供执行后端程序所需的资源。在这种情况下，它将计算任务分布在多个计算实例上，无缝监视和协调其活动，并进行负载平衡。FogBus通过适当的安全功能和容错技术（例如区块链和复制）为此类代理节点提供支持，以便它们在与FGN，FCN和Cloud数据中心进行交互时可以确保可靠性。

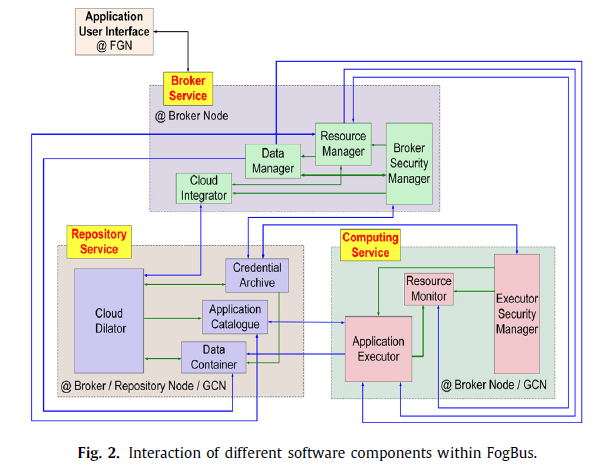
2.通用计算节点（GCN）：出于安全问题，FogBus不会将所有FCN直接暴露给FGN。它们用于一般计算目的，可通过代理节点访问。在这种情况下，代理节点充当GCN的防火墙。此外，代理节点显式管理其资源，并将数据与可执行的后端应用程序一起转发以进行处理。在执行分布式应用程序时，GCN在特定代理节点的监督下形成集群。GCN可以与多个代理节点有关。在这种情况下，基本的矢量时钟用于同步系统（Bravo等，2015）。矢量时钟可帮助GCN识别不同代理节点发给它们的并发命令。以后，并发命令由GCN任意排序，并通知相应的代理节点。在执行命令时，GCN以一对一的方式与关联的代理节点进行交互。此外，为了确保应用程序级别的一致性，GCN一次最多执行一个应用程序。

3.存储库节点：除了进行代理和计算操作外，某些FCN还管理分布式数据库，以促进数据共享，复制，恢复和安全存储。存储库节点提供了用于即时访问和分析历史数据的接口。它们维护各种应用程序的元数据，包括应用程序模型，运行时要求和依赖关系。此外，这些节点可以在应用程序执行期间保留一些中间数据，以便可以从任何异常驱动的停止点开始数据处理。另外，为了确保数据级别的一致性，存储库节点以日志结构化的方式管理所有数据，该方式由其更新时间戳和源驱动。

*云数据中心*：当Fog基础架构过载或服务要求容忍延迟时，FogBus会从Cloud数据中心扩展资源以执行后端IoT应用程序。通过云数据中心，FogBus扩展了全球IoT应用的计算平台。与Fog存储库节点相关联，它可以促进广泛的数据存储和分发，从而使数据的访问和处理变得与位置无关。

**3.2 软件组件**

为了简化IoT-Fog-Cloud的集成，FogBus提供了各种相互关联的软件组件，这些组件可以处理不同硬件工具的操作系统（OS）和对等（P2P）通信级别的异构性。这些组件大致分为三种类型的系统服务。代理服务管理代理节点的功能并根据需要启动其他软件组件，而计算服务负责控制GCN的操作。当代理节点本身开始执行后端应用程序时，将在其中触发计算服务。相反，存储库服务可以在所有Fog节点上运行，以管理与存储库相关的操作。不同的FogBus软件组件之间的交互作用如图2所示。下面讨论FogBus软件组件的详细信息。



3.2.1。经纪服务

Broker Security Manager：从FGN接收到用户的身份验证凭据后，Broker Security Manager会与存储库服务凭据存档相关联地对其进行验证。凭据存档还为该组件提供必需的安全证书，以进行远程云集成。代理安全管理器生成公用和专用密钥值对，以促进端口敲除，特权端口身份验证和基于属性的加密，以保护相应代理节点与其他Fog节点的通信安全。此外，该组件还可以充当区块链接口，以确保在与多个实体交换数据时的完整性。在这种情况下，借助数据管理器，它将根据接收到的数据创建新块。每个块的哈希值和工作量证明被发送到凭据存档，以便在其他节点之间分发，从而可以确保在不同的目的地对链进行一致的验证。代理安全管理器与凭据存档和计算服务的执行器安全管理器一起管理FogBus中的其他安全问题，并为其他组件提供对所需信息的灵活访问。

资源管理器：该组件负责选择合适的资源来执行应用程序。它从存储库服务的应用程序目录中识别不同应用程序的需求，并通过计算服务的资源监控器了解每个代理和GCN中的资源状态。Cloud Integrator协助Resource Manager提供基于云的实例（例如虚拟机和容器）的上下文数据。获得所有信息后，资源管理器会在FCN和Cloud上为应用程序配置所需的资源。在这种情况下，资源管理器会根据其处理能力明确考虑计算实例的异质性。来自FCN和Cloud的计算服务和内部软件系统的Ap​​plication Executor分别帮助Resource Manager提供资源。FogBus有助于服务提供商在资源供应过程中通过Resource Manager应用各种策略。此外，此组件维护一个资源配置文件，该文件可跟踪FCN和Cloud实例以及已部署的应用程序的地址，以便可以将后续数据流直接发送到分配的资源进行处理。该文件还与Cloud共享，用于在相应节点发生故障时恢复放置信息。

数据管理器：此组件从IoT设备接收感测和预处理的数据。它还可以汇总来自多个源的数据，并根据上下文校准数据接收频率。但是，使用此数据将创建块及其链，以与Broker Security Manager关联以维护完整性。稍后，它将数据转发给计算服务的应用执行器进行处理，并以加密方式将其存储在存储库服务的数据容器中，以供进一步使用。在分配的资源上部署应用程序之后，资源管理器将资源配置文件共享给数据管理器，以便它可以直接将后续数据流发送到处理目标。

Cloud Integrator：FogBus与Cloud的所有交互均由Cloud Integrator处理。它将云实例的上下文通知给框架，并将存储和资源供应命令转发到云。通过此组件，FogBus不仅为提供程序提供了用于开发定制的云集成脚本的接口，而且还允许诸如Aneka Calheiros等人的第三方软件系统。（2012b）使用其应用程序编程接口（API）同时处理多个Cloud数据中心。

3.2.2。仓库服务

凭据档案：在IoT设备配置过程中设置的用户身份验证凭据将保留在凭据档案中。它将由Broker Service生成的每个数据块的安全密钥和详细信息分发给其他人。该组件还提供用于云集成的安全套接字层（SSL）和传输层安全性（TLS）证书。此外，它还支持数据容器，用于对存储的数据进行加密和解密。通过存储库服务的Cloud Dilator，它可以定期在Cloud上更新其映像，以便在发生不确定的故障后可以轻松恢复安全属性并在其他属性之间进行分发。

应用程序目录：此组件负责维护各种类型的应用程序的详细信息，包括其操作，从其开发人员推荐的系统属性，执行和编程模型。此外，它指定了资源需求以及应用程序及其成员任务的依赖性。Application Catalog可以通过Cloud Dilator从Cloud扩展此信息。根据其提供的规范，经纪人资源管理器服务条款为应用程序提供资源。根据资源管理器的命令，它还与计算服务的应用程序执行程序关联，在分配的资源上组装应用程序。

数据容器：从物联网设备接收的数据存储在数据容器中，以便可以用于长期分析。在此，通过应用加密技术可确保数据隐私。在应用程序执行期间，它还会从Application Executor接收一些中间数据，这有助于FogBus从任何暂停点重新开始数据处理。此外，在FogBus中，可以根据不同的启用IoT的系统的需求自定义和共享基于数据容器的数据库的架构。此外，数据容器与Cloud Dilator保持同步关联，以掌握远程数据并通过Cloud分散本地数据。

Cloud Dilator：此组件可促进存储库服务的其他软件组件与Cloud进行交互。在这种情况下，Broker的Cloud Integrator服务会帮助Cloud扩张器提供所需的命令，以扩展应用程序规范，传输环安全属性和交换数据。

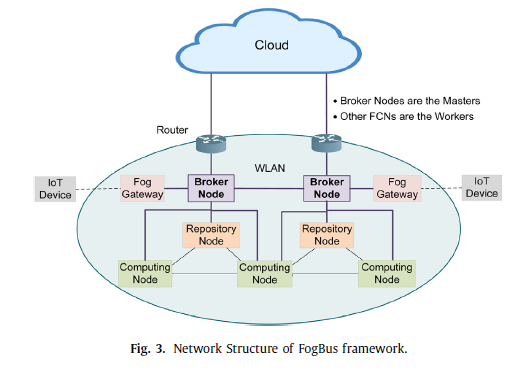
3.2.3。计算服务

Executor安全管理器：执行计算操作时，FCN与其他人的无缝安全交互由Executor安全管理器管理。凭证库的凭证服务可以帮助此组件具有必需的安全属性。该组件与凭证存档和经纪人安全管理器一起，在验证区块链中发挥着重要作用。

资源监视器：此组件与Application Executor一起监视繁忙和空闲状态（例如：CPU使用率，内存占用，网络使用率，功耗等）。基于这些感知的信息，资源管理器为不同的应用程序配置资源。它还在运行时跟踪分配的资源的性能，以满足应用程序的QoS要求。每当资源负载超过服务提供商定义的阈值或发生不确定的故障时，此组件都会立即通知资源管理器。为了应对这种情况，资源管理器可以启动一些操作，例如动态资源供应，应用程序执行迁移和中间数据存储。但是，在当前版本的FogBus中，这些操作以抽象形式保留，以便FogBus用户可以根据系统的上下文自定义其资源管理策略。但是，此组件执行必要的操作以实现系统上的同步。

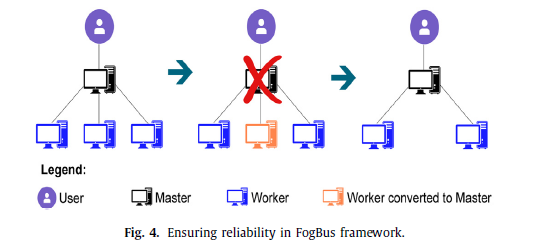
 应用执行器：根据资源管理器发出的配置说明，该组件为相应FCN上的不同应用分配资源。它还从“应用程序目录”扩展了应用程序可执行文件，以便在分配的资源上进行部署。实施应用程序部署后，它将开始接收由Data Manager转发并进行处理的数据。此外，此组件会定期向资源监视器通知资源状态。当检测到或预测到任何异常时，资源管理器会要求此组件从应用程序执行中提取中间数据并将其存储在数据容器中，以使框架具有容错能力。

3.3。网络结构



FogBus的软件组件之间共享大量数据和信息。为了促进它们之间的相互作用，必须在框架的硬件仪器之间进行持久且稳定的网络通信。还需要确保硬件仪器不被通信负担所淹没。另外，FogBus网络应该是安全的，可扩展的并且具有容错能力。认识到这些事实后，我们设计了FogBus网络结构，如图3所示。网络结构的不同方面描述如下。

拓扑：主工作拓扑用于设计FogBus框架的网络结构。在这里，代理节点是主节点，而其他FCN则是工作节点。作为主节点，代理节点从FGN接收数据流和用户信息，并发现用于处理和存储它们的工作程序。在应用程序运行时，它管理工作程序的功能，并将服务结果传递给源自应用程序执行的FGN。此外，它还将支持FogBus的系统的Fog基础架构与云基础架构连接起来。为了促进数据共享并减少主服务器的开销，工作节点还必须在主服务器的明确监督下相互通信。支持FogBus的系统的主机，工作人员和FGN与一个或多个路由器管理的通用无线局域网（WLAN）相关。可扩展性：FogBus框架允许服务提供商根据系统的上下文来扩展活动的Fog节点的数量。连接到同一WLAN的FCN可以简单地通过使其对相应的主设备进行访问而成为工作线程。稍后，主机在该FCN上配置所需的软件组件以执行所需的操作。FogBus支持WLAN中多个主设备的共存，因此FGN可以获得多种选择来分发数据流以进行处理。主人之间也共享工人。在这种情况下，数据完整性和隐私不会受到影响，因为每个主数据库都维护自己的块链和工作人员上独立的数据库。此外，在主服务器上运行的软件组件有助于Fog基础架构同时与多个Cloud数据中心集成。



可靠性：运行多个主机的功能隐式消除了FogBus中主机模型的固有单点故障限制。在应用程序执行和数据管理期间，基于矢量时钟的同步以及一对一的交互支持它们在不同工作节点上各自操作的显式隔离和一致性。该框架允许每个主机在一个工作节点上复制其映像。如图4所示，在该主机不确定的故障期间，此复制有助于相应的工作人员获得主机特权并捍卫通信网络的崩溃。在此，FogBus软件组件的独立于平台的特性起着关键作用。主机还定期检查其工作人员的状态，并将其中间数据和配置（包括已部署的应用程序）存储在不同的位置。当工作人员出现故障时，主服务器与其他工作人员共享该工作人员的信息，以便可以立即开始处理剩余数据。如果主人的所有工人都超负荷，则考虑其他主人的工人。在这种情况下，所有主机之间都保持内部通信。因此，计算工具在框架内始终保持可用。

安全性：如果网络需要适当的身份验证，则在FogBus中包含FGN和FCN。它由管理WLAN的路由器明确处理。管理员还应用网络级别的访问控制和数据包过滤技术来抵抗网络基础设施受到破坏并消除恶意内容。在FogBus中，还存在多个通信链接以到达不同的Fog节点。当发现任何网络异常时，它最终有助于读取路由路径。在与不同的云基础架构进行交互时，FogBus框架中进一步使用了云提供的网络安全策略。

性能：FogBus框架专用于特定系统的网络带宽。由于未与外部实体共享网络资源，因此从网络角度来看，系统的整体性能不会降低。如果服务提供商打算增加框架中FCN的数量，则对额外网络资源的需求将不会很高 高。此外，它还简化了Fog节点的部署，并加快了向用户的服务交付。随着时间的推移，作为本地网络，其吞吐量也保持在可接受的水平。此外，FogBus支持定期调整网络资源，以便它可以处理任何频率和数量的传入流量。此外，FogBus的网络结构在很大程度上不依赖于外部硬件工具来管理和配置网络操作，从而隐式地减少了服务提供商的资本和运营支出。

1. 设计与实现

集成环境的每个硬件都可以支持FogBus中使用的API，执行环境，脚本和编程语言。最终，它可以帮助FogBus超越其OS和P2P通信级别的异构性。以下描述了不同的FogBus元素的实现。

**4.1 系统服务**

FogBus的每个系统服务（代理服务，存储库服务和计算服务）都分为服务接口和管理活动。在主机上，服务接口可帮助从网关设备接收数据和用户规范，并提供服务结果。服务提供商还通过此接口将工作人员的IP地址通知给主服务器。主服务器中的管理活动包含资源供应和负载平衡策略，并更新配置文件。此外，它生成命令并将其转发给工作人员。辅助服务器上的服务接口充当相应节点的接收器，并负责将应用程序的输出文件解码为主机。根据工作人员的命令，工作人员的管理活动可以进行资源分配，监视和状态共享。它还将数据存储在关系数据库中，并为应用程序的后端程序创建输入文件。除了区块链，不同系统服务的服务接口和管理活动还共同处理其他安全方面的问题，例如对FogBus中的主服务器和工作服务器进行加密和身份验证。在FogBus中，每个系统服务的服务接口都是作为Web程序实现的。它们基于PHP（一种HTML嵌入的服务器端脚本语言）开发，并使用基于HTTP协议的RESTful API在WLAN中的不同FCN之间交换数据和共享信息。基于PHP的Web程序可以在每个操作系统中运行，例如Unix，Windows，Linux和NetWare。另一方面，HTTP是一种应用程序层协议，可以适合与不同的传输层协议（例如TCP和UDP）一起运行。此外，大多数嵌入式，网络和物联网设备都设计有用于HTTP通信的内置协议栈，或者支持其轻松安装。因此，FogBus系统服务的服务接口可以跨不同类型的OS和P2P通信标准运行。在FogBus中，每个FCN中还部署了一个Apache服务器，以运行相应的系统服务的Web程序。此外，在FogBus中，每个系统服务的管理活动都是使用Java编程语言开发的。编译的Java程序在Java虚拟机（JVM）上运行，可以在各种OS上轻松安装。因此，FogBus系统服务的管理活动可在多种平台上运行。另外，MySQL服务器安装在FogBus的不同FCN中，以管理数据库及其操作。

**4.2 区块链**

维护数据的完整性并确保未通过未注册的源发送数据对于系统的信誉非常重要。为了防止数据完整性和数据篡改，Zyskind等人最近在许多实时系统中采用了区块链技术。（2015）。从理论上讲，区块链是一组分布式账本，可以对它们进行编程以记录和跟踪任何事物的价值。在区块链中，只要分布式系统的实体接收到新数据，就会将数据形成一个块。该块拥有一个哈希值，通常通过使用相应的数据，该块在链中的索引，数据接收的时间戳以及该链中其前一个块的哈希来创建哈希值。此外，该节点与链中的其他区块一起挖掘该区块，以为该区块创建工作量证明，以便其哈希遵循与其他区块相似的模式。之后，将数据，块的副本发送到其他节点，以与其本地链链接。在此操作中，节点对块进行挖掘以证明工作量证明。数字签名也用于验证目标块的来源。但是，如果在一个节点上更改了任何块的数据，则该块的哈希将更改，并且与保存在下一个块中的哈希不匹配。结果，链的后半部分将失效。为了使链再次有效，需要重新计算无效块的哈希值。此外，需要再次生成每个块的工作量证明。两种操作都非常耗时且计算量大。此外，除非通过遵循同一组操作分别重构其分布式副本的50％，否则对区块链中数据的这种欺诈性操作将不会成功。因此，在严格的时间限制内更改区块链中的任何数据变得非常困难（Swan，2015）。在FogBus中，主控器使用SHA256算法根据接收到的数据创建数据块，并根据数据，前一个数据块的哈希，时间戳和随机数计算每个数据块的哈希值（Brownworth，2017）。管理员还创建随机的公共/专用密钥对，以帮助使用原始数据生成唯一的签名。后来，他们与工作人员共享Blockchain详细信息，数字签名属性和Base64编码格式的数据。利用接收到的母版公钥，工作人员可以验证数据是否来自合法来源。如果使用其他任何键，则该数据将被拒绝。在这种情况下，公私钥对在每个块中保持动态，以防止使用蛮力技术生成私钥。此外，通过挖掘支持工作量证明的现时值，在工作人员处验证每个块及其散列。如果任何工作人员报告有关区块链篡改或签名伪造的错误，则将大部分网络中的区块链复制到该节点。FogBus还通过在每个工作人员上显示区块链副本的最新哈希值，向用户和提供者提供跟踪通过主服务器上运行的服务接口的数据/块流的信息。因此，它可以帮助用户和提供商对FogBus网络中的可疑活动采取必要的措施。在FogBus中，区块链是用Java编程语言开发的，并且在FogBus的不同FCN中，该实用程序直接与相应的系统服务进行交互。

**4.3 云插件**

在FogBus中，在主服务器上运行的服务接口会提示用户指定有关云集成以进行数据处理的意图。如果用户希望扩展Cloud资源进行计算，则只有在主服务器上部署的FogBus Cloud Plugin才被激活。对于其他操作（例如存储和分发），主服务器上的“管理活动”直接与云进行通信。但是，FogBus为提供商提供了灵活性，使其可以使用不同的定制或第三方云插件服务来集成Cloud和Fog基础架构以进行计算。如果运行第三方Cloud Plugin服务，则根据该插件的要求配置主服务器。但是，要开发定制的插件，最好使用跨平台编程语言。在当前版本的FogBus（Aneka）中，第三方软件用于Cloud集成以执行计算操作。Aneka是一个PaaS框架，用于促进基于云的应用程序的管理（Calheiros等，2012b）。Aneka框架以面向服务的方式运行。它配备了一套软件组件，用于配置，操作和监控Aneka-Cloud环境。Aneka-Cloud可以由来自公共或私有或混合云的异构实例组成。Aneka为开发人员提供了多种API，用于在Aneka-Cloud中提供和调度物理和虚拟资源。开发人员使用不同的编程模型来制定应用程序的逻辑，并为它们的部署和执行设置运行时环境。目前，Aneka平台支持任务包，分布式线程，MapReduce和参数扫描模型。在FogBus的基于Aneka的Cloud插件中，Cloud实例的IP地址由提供商指定。该插件可以在Aneka-Cloud中启动任务和线程模型，以分别在单个和多个Cloud实例上进行数据处理（Calheiros等，2012a）。根据FogBus的内置资源供应策略，首先使用Fog基础结构来处理数据。如果Fog基础架构上的负载增加到阈值，则该应用程序及其输入数据流将被称为Cloud基础架构。对于FogBus中的第二种情况，主要的“管理活动”将数据存储在Cloud输入文件中。主服务器上基于Aneka的Cloud插件每500毫秒的轮询周期就会对此文件进行解析，并检查待处理的数据以进行处理。如果存在任何待处理的数据，则它将形成任务或线程；在Aneka-Cloud上封装数据并启动到一个或多个Cloud实例。在这种情况下，还应使用区块链来确保数据完整性。

**4.4 应用**

FogBus支持执行和部署不同的启用IoT的系统的应用程序。在FogBus中，这些应用程序分为前端程序和后端程序。尽管应用程序不是FogBus软件组件的一部分，但是FogBus为开发人员提供了一些指导方针，以构建符合FogBus框架功能的前端和后端程序。应用程序的前端和后端程序所需的规范描述如下。

4.4.1前端程序

应用程序的前端程序在FGN中运行。大多数FGN的底层平台是Android，iOS，Windows，Tizen，WebOS和RTOS。在这种情况下，这些平台应支持用于开发前端程序的编程语言。此外，对于某些应用程序，前端程序需要临时存储数据。关于这一点，开发人员应在这些平台上使用兼容的数据库系统和架构。此外，前端程序还处理来自物联网设备的传入数据。大部分IoT设备都运行Bluetooth Low Energy网络技术进行通信，因为它们受到了能源的限制。要处理此问题，前端程序应支持常规和低能耗蓝牙交互。在FogBus框架中，前端程序与主机上运行的服务接口直接相关，以转发IoT数据和用户信息，并接收服务结果。为了简化这些交互，前端程序的用户界面可以设计为易于解析主服务界面的Web程序的方式。

4.4.2后端程序

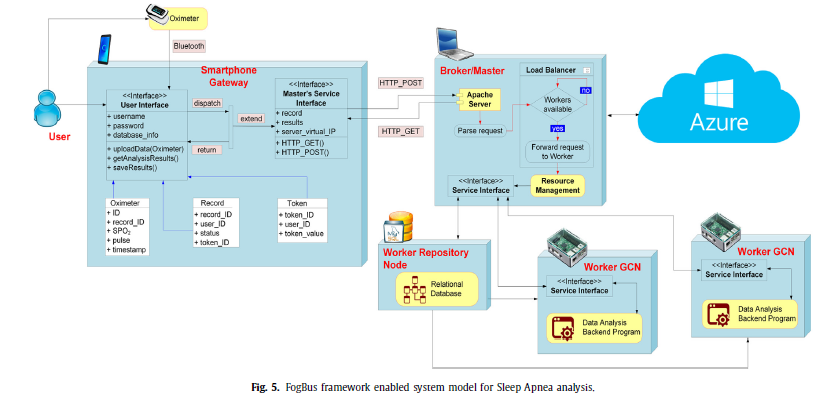
在FogBus中，应用程序的后端程序在FCN中执行。由于FCN是分布式的，因此要充分利用其功能，最好以分布式方式构建后端程序。在这种情况下，开发人员可以应用后端程序的模块化开发。此外，后端程序的执行不应受到FCN操作系统级别的异构性的阻碍。为了解决此问题，开发人员可以使用跨平台编程语言（例如Java）来开发后端程序。在开发后端程序时，应在脚本中指定一些特定点，以便可以在执行期间存储应用程序在这些点上的中间数据。此外，后端程序应能够提取输入文件并在工作人员处更新输出文件。

1. 案例研究：睡眠呼吸暂停分析

在这项工作中，采用了FogBus框架来部署和执行名为“睡眠呼吸暂停分析”的实际应用程序。睡眠呼吸暂停是一种在睡眠过程中空气停止流入肺部10秒钟或更长时间的疾病。因此，它降低了患者血液中的氧气含量，降低了心跳速率，类似于患者停止了呼吸。它可能非常频繁地发生，并严重阻塞患者的声音睡眠。此外，如果老年和哮喘患者的血氧饱和度明显降低，睡眠呼吸暂停可能会引发心力衰竭或脑中风。但是，睡眠呼吸暂停是一种非常常见的疾病，尽管大多数人都忽略或不知道。为了确定睡眠呼吸暂停的强度，需要不时监测血液中的氧饱和度。如果强度高于正常水平，建议在发生其他并发症之前咨询医生（He等，1988）。通常，睡眠呼吸暂停分析既困难又麻烦，因为需要进行通宵睡眠研究才能掌握必要的数据。在此过程中，脉搏血氧仪和心电图（ECG）机在睡眠期间会与患者身体的各个部位相连。根据接收到的周围毛细血管氧饱和度，SpO2和ECG数据，医生确定以比例方式显示睡眠呼吸暂停强度的患者的呼吸暂停低通气指数（AHI）。当前，要进行睡眠呼吸暂停分析，需要医院或实验室用的设备，而这些设备对于个人而言昂贵。此外，在对关键患者进行监视的同时，此分析变得非常敏感。因此，我们开发了一种使用FogBus框架进行低成本睡眠呼吸暂停分析的原型，该原型从手指脉搏血氧仪收集SpO2和心跳率（Nigro等，2011），并利用当地资源进行处理。与基于云的处理相比，它为患者提供了可负担的，易于配置的方法，并提供了更快的结果。启用FogBus的睡眠呼吸暂停分析原型的详细信息如下所述。

**5.1 系统配置**

启用FogBus的睡眠呼吸暂停分析原型的系统设置如图5所示。下面给出了不同硬件仪器的配置。



物联网设备：跳线JPD-500F手指脉搏血氧仪，1.5V，蓝牙低功耗v4.2（BLE），UTF-8数据编码。

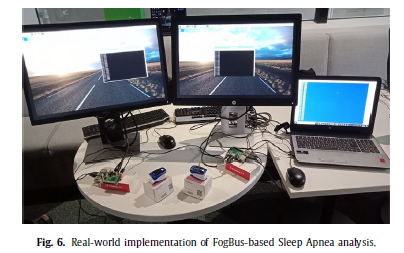
网关：智能手机，Oppo A73 CPH1725，Android 7.1.1。

代理/主节点：Dell Latitude D630笔记本电脑，Intel（R）Core®2Duo CPU E6550 @ 2.33GHz 2GB DDR2 RAM，32位，Windows 7，Apache HTTP Server 2.4.34，Java SE运行时环境（ JRE）1.6，MySQL 5.6，.net 3.5，Aneka 3.1。

  其他FCN / Worker节点：Raspberry Pi 3 B +，ARM Cortex-A53四核SoC CPU @ 1.4GHz 1GB LPDDR2 SDRAM，IEEE 802.11,64位，Raspbian Stretch，Apache HTTP Server 2.4.34，JRE 1.6，MySQL 5.6。

公共云：Microsoft Azure B1s 机器，1vCPU，1GB RAM，2GB SSD，Windows Server 2016，.NET 3.5，Aneka 3.1。

图6描述了该系统模型的实际实现。



**5.2 已安装的套件**

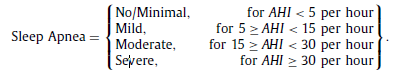
用于睡眠呼吸暂停分析的已开发原型主要以Fog基础设施为中心。但是，如果Fog基础结构无法处理数据，则使用FogBus内置的基于Aneka的内置Cloud Plugin，将数据发送到Azure VM。原型中安装的用于睡眠呼吸暂停分析的应用程序包包括一个肛门前端和一个数据分析后端程序。下面给出了已安装软件包的说明。

5.2.1智能手机网关上的Android接口

名为HealthKeeper的android可执行文件将启动与原型操作员的android接口。安装在智能手机上的可执行文件使设备可以充当脉搏血氧仪和主机之间的中介者。它是在MIT App Inventor（一个开源平台）上开发的（实验室，2015年）。该界面分为“主页”和“会话”屏幕（图7）。主页屏幕可帮助操作员将血氧仪与智能手机配对，以使用蓝牙接收患者数据，并输入主机的IP地址。会话屏幕处理与主机的所有交互，包括数据传输。在这种情况下，界面会自动记录和发送数据，而不是通过HTML表单手动发送数据。录制开始时，将初始化一个空数据列表并重置计时器。从血氧仪收到的每个数据值都将附加到列表中。停止记录后，该列表将发送给主服务器，以供存储和分发给工作人员。该屏幕还扩展了在主服务器上运行的服务接口，并在操作员可将其显示给主服务器后将结果显示给操作员。

5.2.2。工作者计算节点上的数据分析

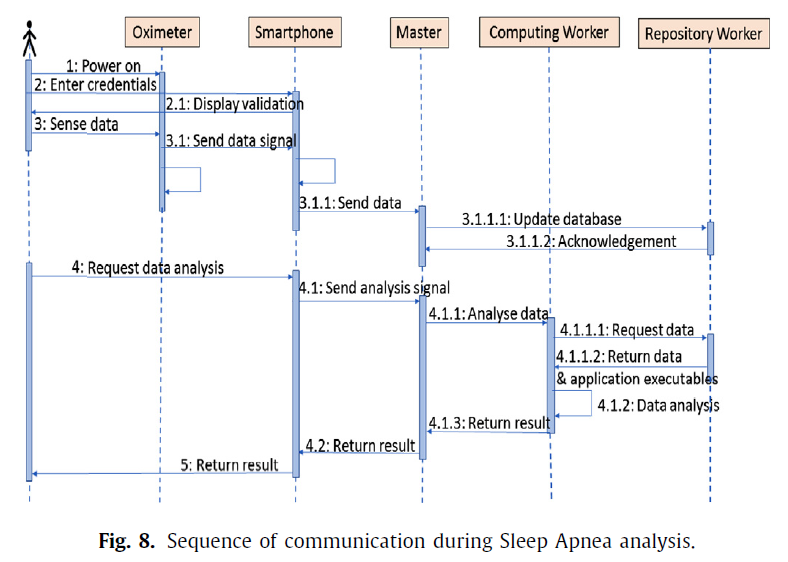
睡眠呼吸暂停分析的数据分析封装了Manigadde（2018）和Initiative（2017）中发现的两个开源程序。这些Java程序存储在存储库工作程序中，并根据主服务器的命令将它们转发给计算工作程序进行安装。数据分析将输入数据作为文件。从输入文件中，将第一，第二和第三列分别解析为时间戳，心跳率和血氧水平。在分析中，布尔变量跟踪氧水平是否下降。每当氧气水平低于88时，浸入布尔变量将变为true并保持为真，直到氧气水平高于88。这通过氧气水平浸入发生的附近时间戳中心跳率的升高得到验证。分析中的计数器变量说明dip布尔变量已更改为true的次数。此计数称为呼吸暂停-低通气指数（AHI），用于确定睡眠呼吸暂停的强度。下面给出了基于AHI的睡眠呼吸暂停分析案例。



但是，作为附加信息，数据分析会存储给定时间段内的最低氧气水平。对于心率数据，将标识最小值和最大值。还确定平均心率和平均心率的上升或下降。另外，在氧水平下降期间的心跳模式被过滤并产生ECG。识别出这些信息和睡眠呼吸暂停强度后，分析结果将结果存储在文件中。该文件稍后由主服务器的服务接口解析，以通知原型操作员。

**5.3通讯顺序**

在支持FogBus的睡眠呼吸暂停分析的原型中，所有硬件仪器都属于同一WLAN。它们的通信顺序如图8所示。该序列通过使用操作员所需的凭据通过智能手机配置脉搏血氧仪来启动。血氧仪可感测患者的SpO2和心跳速率，并通过蓝牙通信转发至智能手机。这些数据从智能手机发送到主服务器。主服务器稍后将数据存储在存储库工作器中。从存储库工作器到主服务器的存储操作确认确认之后。由于智能手机扩展了主机的服务接口，因此操作员可以看到此确认。在记录了特定时间段的数据后，操作员通过智能手机向主机提示请求，以分析存储的数据。然后，主机与合适的计算机工作者进行通信，并向其发出数据分析所需的特权。计算工作者从存储库工作者请求存储的数据和分析可执行文件。收到这些元素后，计算人员将开始分析操作。分析操作完成后，结果将发送回主机。智能手机从主机提取结果并将其显示给操作员。



1. 框架特征评估

**6.1 实验装置**

为了评估FogBus的框架特性，我们通过以下方法实现了Stack4Things（Bruneo等人，2016），基于Cloudlet的PaaS（Yi等人，2015a）和Indie Fog（Chang等人，2017）框架的不同功能。遵循给定的准则，并与FogBus进行比较。在实验期间，FogBus的区块链功能保持启用状态，并且该框架管理集成的Fog-Cloud基础架构。在实验中，数据处理请求无间隔地顺序发送到框架。在此，评估以下框架特征。

1.资源负载：框架的计算处理器和内存使用情况可以指示其资源负载。由于大多数Fog节点的资源并不丰富，因此执行重量级软件系统可能会导致大量计算开销。因此，需要在Fog计算环境中部署轻量级框架。消耗较少计算处理器和内存的框架被认为比其他框架更轻。

2. QoS期望丢失率：由于管理开销和资源限制，框架通常无法满足用户的QoS期望。较低的QoS预期未命中率反映出框架在管理和利用其计算资源方面的更高效率。

3.基于时间的属性：系统启动时间和数据检索时间对于指示框架的性能至关重要。互连和启动框架中所有组件所需的时间由系统启动时间定义。数据检索时间是指通过框架访问数据所需的时间。这些基于时间的属性的值越低，表示框架的响应性越高。

**6.2 结果分析**

6.2.1 框架轻巧

除FogBus外，其他框架还广泛集成了第三方软件系统。结果，它们在核心/主服务器/代理Fog节点的计算（CPU）和内存（RAM）资源上的负载要高于FogBus（图9）。此外，在基于Cloudlet的PaaS框架中，系统操作不会分布在多个Fog节点上，这会增加核心Fog节点的CPU和RAM利用率。相反，在Indie Fog和Stack4Things中，尽管可以协同操作Fog节点，但是Indie Fog注册表和在核心Fog节点中运行的虚拟板还具有其他职责，例如联合资源，存储库和安全性管理，服务发现，应用程序服务管理 结果，Indie Fog和Stack4Things框架的核心Fog节点上的CPU和RAM负载增加。

* + 1. 管理资源的性能

FogBus框架能够同时利用边缘和远程资源。当Fog节点过载时，FogBus将继续通过基于Cloud数据中心的资源进行输入处理。因此，它提供了更大的范围来满足用户在处理其数据方面的QoS期望。结果，Fog-Bus中的QoS预期未命中率变得比InDie Fog相对要低，在InDie Fog中，应用程序的执行仅由基于Fog节点的资源来管理（图10）。在基于Cloudlet的PaaS框架中，CPU和RAM上的过多负载迫使核心Fog节点放慢了对输入数据流的管理操作，与轻量级FogBus相比，这增加了其QoS期望缺失率。此外，由于通过Cloudlet中部署的核心Fog节点管理应用程序服务，因此Stack4Things上的QoS预期未命中率略高于FogBus。

6.2.3 框架的响应能力

  与其他框架相比，FogBus的系统启动时间更短（图11）。在这种情况下，与垂直帮助FogBus相比，更少的第三方软件依赖项，按需云交互以及更多的水平连接。FogBus中的数据检索也比其他时间花费更少的时间。FogBus以分布式方式将数据存储在本地存储库节点中，而不是将其发送到Cloud或任何集中式数据存储。因此，在应用程序执行期间，可以通过FogBus轻松访问数据。

1. 应用部署评估

FogBus支持各种资源管理和调度策略的实施，以执行使用并发编程模型（例如SPMD（单程序和多数据），工作流和流）组成的IoT应用程序。作为示例说明者，我们选择了使用SPMD模型开发的医疗应用程序，以评估FogBus中的应用程序部署方案。

**7.1实验装置**

第5节中讨论的用于睡眠呼吸暂停分析的原型用于在执行应用程序时评估FogBus的区块链和管理开销对各种系统参数（例如服务等待时间，能耗和网络使用率）的影响。为了进行实验，在特定时间段内记录了多种肟的数据，随后主设备对每个记录的数据块依次生成处理请求，以计算人员。FogBus中的每个实验方案均在以下设置下建模。

1.有/无间隔：在有间隔设置中，主机在接收到前一个请求的结果的一定间隔后，将下一个数据处理请求发送给其计算人员。此时间间隔有助于主机和计算人员减少开销。相反，在“无间隔”情况下，主服务器在先前请求的结果可用时立即将下一个请求发送到其计算工作者。它确保F​​ogBus框架始终保持活动状态，并且节点上没有空闲时间。

2.有/没有区块链：FogBus提供了灵活性，可以根据用户和服务提供商的要求启用或禁用其区块链安全功能。根据FogBus中区块链功能的状态，此实验设置的各个部分彼此不同。

3. Fog /仅云/集成：FogBus支持跨各种计算基础架构的应用程序执行。该实验设置是指应用程序执行是仅在Fog或Cloud还是集成基础架构上执行。在实验过程中，在Master和Azure VM上使用Microsoft Performance Monitor记录了数据参数，而在Raspberry Pi电路上则使用了NMON Performance Monitor（Microsoft，2017; Splunkbase，2018）。除了第5.1节中指定的系统模型参数外，表2中还提供了用于实验的其他参数。

**7.2 结果分析**

7.2.1请求数

图12描绘了在FogBus中不同实验设置下生成的请求数。可以看出，与仅云和集成雾云情况相比，“仅雾”设置中的请求数量更高。之所以会这样，是因为Fog基础设施可以快速交付上一个请求的结果。在“无间隔”设置期间，此值比“有间隔”设置显着增加，因为请求是由主机连续生成的。还要注意的是，如果关闭FogBus的区块链功能，则会生成相对更多的请求。在这种情况下，由于通过基础结构共享和处理的附加数据量较少，因此可以提高接收先前请求结果的速度。根据这些观察，可以理解，如果存在更多的请求而安全性要求较低，则可以将FogBus设置为“ Fog Only”（禁用雾），并禁用区块链功能。但是，在这种状态下，基础结构的管理和处理开销将随请求数量和单个请求的数据块大小成比例增加。可以通过调整请求创建之间的间隔进行管理。

7.2.2服务延迟

图13展示了FogBus不同设置对服务等待时间的影响。这里，服务等待时间被建模为网络传播延迟与任务完成或应用程序执行时间的总和。众所周知，Fog基础架构的计算能力并未得到增强，但它更靠近数据源。因此，对于Fog基础架构，网络传播延迟要小得多。此外，如果请求的数据块的大小不是很大，则无论应用程序是在Fog还是Cloud中执行，其完成时间都不会有很大不同。由于在此实验中，用于请求的数据块的大小不是很大，因此服务交付延迟很大程度上取决于网络传播延迟。结果，在“仅雾”的FogBus设置中，与“仅云”和“集成雾-云”情况相比，服务交付延迟最小。由于区块链功能的管理增加了更多时间来完成处理请求，因此此延迟在禁用状态下变得更低。此外，在这种情况下，“使用间隔”设置可以减少基础结构和网络的开销。这也有助于改善服务交付延迟。因此，可以认识到，这些设置有助于FogBus处理具有严格期限的请求。

7.2.3 网络使用

图14显示了FogBus不同设置下的网络使用情况。在此实验中，“仅雾”设置比“仅云”和“集成雾-云”情况提供了更高的性能，因为它仅利用本地网络资源。禁用的区块链功能还可以减少网络使用量，因为需要在基础架构之间传输较少的安全属性。但是，当连续生成请求并交换其关联的数据和信息时，网络使用率会提高。在这种情况下，调整后续请求生成间隔可以将网络使用降低到一定程度。因此，即使为支持IoT的系统分配了较少的网络资源，这些调整也使FogBus正常运行。

7.2.4能源

图15显示了不同的FogBus设置如何影响基础设施的能耗。在“仅Cloud”设置中，将Fog节点用于联网，而Cloud VM进行计算，而在“仅Fog”设置中，联网和计算均由Fog节点处理。在集成雾云的情况下，根据系统的上下文将计算任务分配给两个基础架构。由于与Fog节点相比，云VM消耗的能量更多，因此在Fog中，只需设置较少的能量即可进行操作。此外，为了管理FogBus的区块链功能，还需要开发更多的能源。在这种情况下，禁用的区块链功能可为FogBus节省一些能量。另外，与空闲时间相比，繁忙时间的基础结构能耗更高。因此，后续请求创建之间的间隔有助于改善基础架构的能耗。但是，它导致FogBus处理的请求数量较少，可以通过有效地调整间隔时间来解决。这些配置可帮助FogBus在能源紧张的情况下执行应用程序。

1. 结论与未来工作

在这项工作中，我们提出了FogBus框架，该框架可以将不同的支持IoT的系统集成到Fog和云基础架构中。该框架促进了物联网应用的部署，资源监控和管理。FogBus的系统服务是以跨平台编程语言（PHP和Java）开发的，并与可扩展的应用程序层协议（HTTP）一起使用，可帮助FogBus克服不同Fog节点的OS和P2P通信级别的异构性。此外，FogBus框架还充当集成式雾云环境的平台即服务（PaaS）模型，不仅可帮助应用程序开发人员构建不同类型的IoT应用程序，而且还支持用户定制服务，服务提供商根据系统的上下文来管理资源。由于某些启用了IoT的系统（例如，健康监控和公用事业服务计量）处理敏感数据，因此FogBus将身份验证用于数据隐私，将区块链应用于数据完整性。为了在不太安全的网络上进行数据传输，在FogBus中应用了加密技术。根据FogBus的原理，还开发了一种经济高效的睡眠呼吸暂停分析原型。此外，与现有框架相比，我们认识到FogBus的软件组件是轻量级的，具有足够的响应能力，并且能够利用边缘和远程资源。此外，不同的FogBUs设置可以在执行应用程序时有效地应对多种情况。尽管FogBus能够提高跨各种基础架构的服务质量，但仍可以在以下方面在更大范围内进行改进。

资源管理策略：FogBus可以灵活地应用定制的供应策略，同时为不同的应用程序分配资源。可以针对现有的静态管理策略开发动态资源管理策略，以实现计算基础架构之间的负载平衡和QoS增强。

雾基础设施虚拟化：FogBus协助将雾和云计算与具有IoT的系统进行集成。尽管可以对云计算进行虚拟化，但仍需要深入探索以虚拟化FogBus中的Fog基础架构。

人工智能：目前，FogBus不支持任何人工智能技术来控制不同基础架构中的操作并提高系统的弹性。包含人工智能技术可以对FogBus做出重大贡献。

应用程序放置技术：FogBus本质上支持分布式应用程序执行。在以分布式方式放置应用程序时，服务等待时间，用户期望和部署成本成为主要问题。在这种情况下，可以将不同的有效应用程序放置技术添加到FogBus的软件堆栈中。此外，将来可以对FogBus进行一些比较性能研究，以处理不同的计算密集型，网络密集型和存储密集型应用程序。

运行时应用程序迁移：如果预测到任何异常，则在运行时迁移应用程序至关重要。可以针对FogBus开发不同的运行时应用程序迁移策略来处理此类不确定事件。

 轻量级安全功能：FogBus的现有安全功能需要相对较高的计算帮助。因此，这会影响服务交付延迟，能耗和网络使用率。因此，轻便而有效的安全功能有助于进一步提升FogBus。

嵌入式Blcokchain功能的改进：FogBus的嵌入式Blockchain功能是通用的。但是，它提供了抽象，因此可以在减少数据检索时间，管理智能合约以及在集成雾云环境中实施链链的未来研究中发挥重要作用。

软件可用性

我们发布了FogBus作为开源软件。可以从https://github.com/Cloudslab/FogBus访问其源代码以及用户和开发人员手册。