基于以太坊区块链的物联前端应用开发

摘要

有几种分布式账本协议可能适用于物联网（IoT），包括以太坊，Hyperledger（超级账本）Fabric 和 IOTA。本文从物联网应用程序开发的角度简要介绍和比较它们。基于区块链（BC）的物联网应用可以集成链上逻辑：智能合约，以 Web，移动或嵌入式客户端前端应用部分。我们为 IoT 前端 BC 应用程序提出了三种可能的体系结构。它们在以太坊区块链客户端（本地设备，远程服务器）的定位和用于管理传出事务所需的密钥存储的定位方面有所不同。这些利用以太坊网络进行可信交易交换的体系结构的实际限制是数据量，完整区块链节点的位置和同步以及位置和对以太坊密钥库的访问。这些实验的结果表明，完整的以太坊节点不太可能在受约束的物联网设备上可靠地运行。因此，具有远程以太坊客户端的体系结构似乎是一种可行的方法，其中存在两个子选项并且在关键存储位置/管理方面有所不同。另外，我们提出使用物联网设备和远程区块链客户端之间专有通信的体系结构，以进一步减少网络流量并增强安全性。我们预计它也能够在低功耗，低比特率的移动技术上运行。我们的研究阐明了架构方法的差异，但特定总账协议和前端应用程序架构的最终决策强烈基于特定的预期用例。

2017 年国际物联网识别，信息和知识会议（IIKI2017）科学委员会负责选择和同行评审。

关键词：架构;区块链;以太坊; 前端应用程序;物联网

*作者联系方式 电话：+386 1 4768844.电子邮件地址：matevz.pustisek@fe.uni-lj.si

2017 年国际物联网识别，信息和知识会议（IIKI2017）科学委员会负责选择和同行评审。
10.1016 / j.procs.2018.03.017

1.介绍

我们可以观察到联合使用物联网（IoT）和分布式账本技术的初始尝试。这些尝试倾向于研究这种应用程序开发方法的可行性，提供概念验证（PoC）并探索可能的商业机会。 物联网[1]是一个成熟的概念，涉及众多相互关联的事物以及相应的基于云或雾的应用程序。

它正在革新互联网，并被部署在各种应用领域。 另一方面，分布式账本目前主要通过区块链技术（BC）实现 - 仍然在兴起[2]。 尽管如此，它们很可能会像物联网一样破坏 ICT 系统，服务和应用领域。

现有的区块链系统的范围在技术特征方面以及在用户和开发者社区中的接受度方面不同。 通过基于区块链的物联网解决方案部署的第一个示例，当前区块链设计中的某些低效率开始显现。 由于高额交易费用和长时间的交易确认时间，微支付在比特币网络中变得几乎不切实际。 由于区块链的大小和有限的事务处理速度，物联网（预计数十亿台设备）所需的可扩展性往往受到限制。 现有的 BC 协议通过功能扩展尝试面对这些低效率问题。 与此同时，正在开发新的分类账协议，从头开始内置 IoT 需求。

物联网和区块链这两个发展自然都在寻求将其融合在一起，从而为应用开发和使用提供了巨大的空间。 然而正确的方法和选择适当的技术远非直截了当。 它关键取决于预期用例的细节。 预见使用看似微小的变化可能会导致复杂性的急剧增加以及适应解决方案的额外努力，甚至可能是不可能的。

我们研究的目标是分析和展示基于以太坊（ETH）BC 的物联网应用开发的实际限制。 因此，我们详细阐述并比较了基于 ETH BC 的前端物联网设备应用设计的架构方法。 我们实现了这些体系结构的三个版本，并在性能和安全性方面进行评估。 该研究为 IoT 应用程序开发人员提供指导，帮助他们选择适当的系统设计并避免对 IoT 设备和 BC 技术施加不切实际的期望。 他们的架构方法可以根据计划的用途和计划的物联网系统的具体情况进行调整。

在第 2 节中，我们简要介绍了现有技术状况，包括目前作为物联网区块链技术的可行候选人以及在物联网中使用区块链的一些情况的三种分布式总账协议。 在第 3 节中，我们概述了物联网区块链申请开发的原则。 第 4 部分介绍并比较了区块链启用物联网设备的四种不同架构方法，并分析了它们的正面和负面，从我们的实际实验中得出。 第 5 部分总结了该文件，反映了区块链在物联网方面的未来发展。

2.最顶尖的技术

区块链技术是众所周知的加密货币基础，但也提供了许多其他可能的应用领域。 BC 有两个关键的应用领域，具有不同的业务需求[3]：金融科技（FinTech）和物联网。尽管这两个领域都需要交易的分散式可信分类帐的基本共同特征，但是在例如原则用途，交易量和交易比率，安全要求的严格程度或交易成本。在金融科技领域，主要挑战是确保绝对安全和值得信赖的金融支付，交易量低，对交易延迟容忍一些。另一方面，在物联网中，预计会有大量设备和大量交易量，并需要微型和纳米支付才能实现物联网资产和数据货币化。交易成本在这里成为一个相关问题，以及近乎实时操作所需的交易延迟。

物联网 BC 解决方案的成功使用案例并不多，这些解决方案超越了简单的概念验证（PoC）或高于技术准备水平（TRL）4，并且不仅仅包含几个设备。 这并不奇怪，因为区块链物联

网应用领域仍处于起步阶段。 这些活动主要是为了澄清不列颠哥伦比亚省在物联网中的作用，测试实施中的局限性并探索可能的商业机会。 尽管如此，已经提出了一些有趣的用例，主要是在智能电网和电费，物流和物联网设备管理领域。

在智能电网中，目前 IoT 和 BC 正在解决的关键挑战是智能抄表，在当地微网中销售剩余能量，电动汽车充电和需求侧管理[4]，[5]。在物流方面，物联网和区块链正在对产品识别和货物跟踪进行调查。在[6]中提出了一种解决方案来追踪容器，这些容器测量光线，温度和其他环境参数，然后在区块链中保护这些信息。这对于证明货物符合食品或医疗产品规定是非常重要的，或者如果不符合运输条件，甚至可以自动收取罚款。在[7]中，类似的方法被应用于制药供应链。 Zerado 致力于基于 NFC 和 BC 的房地产访问控制[8]，以实现共享经济的增长。物联网设备管理是其他应用领域的基础，因为它包括访问和存储在区块链中的物联网数据。在[9]中，这个概念在智能家居场景中被证明可以管理家用电器和电力消耗。 [10]对自动售货机的管理也有类似的想法。

2.1 分散的分布式账本技术

分布式分类账的各种规格和实现都是可用的，但在我们看来，目前有三种分布式物联网具有相关的前景。 这是两个区块链，以太坊 [11]和超级账本 Fabric [12]。 第三个是 IOTA [13]，它基于新的无块分布式分类账架构。 尽管比特币（BTC）[14]可能是最着名的区块链协议，它主要是由于流行的加密货币比特币而获得声誉，但它在物联网和/或分布式应用程序开发中的潜在作用是非常有限的，并不是一个可行的物联网区块链解决方案。 比特币协议即缺乏分布式链上智能应用，链路规模很大，交易确认延误时间长。 因此，它的作用或多或少地局限于支金融科技应用程序中的加密货币。

表 1.物联网的分布式账本技术比较

	比特币	以太坊	超级账本 Fabric	IOTA
原生加密货币	是	是	否	是
去中心化应用	否(非常有限)	是 – 智能合约: Solidity 语言实现	是– 链码: 使用 Go, Java 语言实现 - (在容器中执行)	否 (非常有限)
交易手续费	有	有	无	无
网络类型	共有	共有(或 私有)	私有	共有
网络地址	无需认证	无需认证	需认证	两者都有
账户匿名性	是	是	否	两者都有
状态通道	闪电网络	雷电网络、分片	不需要	不需要
物联网适用性	否	是(有一些局限)	是	是
去中心化应用适用性	否	是	是	否

在区块链的新应用领域，例如 IoT，流行的区块链协议中的其他实际限制正在变得明显。现有的区块链协议试图应对增加的限制，或多或少地成功修补核心区块链协议。例如，状态通道将离线和在线交易结合起来，与当前区块链体系结构相比，有助于实现额外的可扩展性，隐私性和减少确认延迟。在以太坊中，这种方法表现在雷电[15]和闪电网络中的比特币中[16]。以太坊智能合约无法联系外部 URL，这限制了它们与“链外的世界”的整合。这个缺点可能会被神谕者所超越[17]。它们充当中介，提供数据馈送以及区块链表/外部软件（例如网站）或硬件实体的真实性证明。这些附加软件已经引起了一些兴趣，但尚不成熟（例如，已公布的路线图与实际交付日期之间的强烈不匹配），而且几乎没有实际的接受程度。这就解释了为什么 IOTA 采取了不同的方法，从一开始就为物联网设计了分类账技术（及其周围的整个系统）。

在表 1 中比较了三种分布式账本技术。以下小节将更详细地介绍特定技术。

2.1.1 以太坊

以太坊白皮书[11]是描述以太坊的初始文件，它解释说以太坊协议最初被设想为一种加密货币的升级版本，提供诸如 onblockchain 托管，提款限额，金融合同，赌博市场和就像通过高度泛化的编程语言一样。以太坊基金会正在开发的以太坊协议在黄皮书[18]中有详细说明。

ETH 协议是 BC 协议。新事务形成由挖掘节点验证的块。矿工使用工作证明（PoW）一致性算法。矿工因采矿费而受到奖励，由交易发行人支付。 ETH 节点可以参与两个公共网络之一，即主网络或测试网络 Ropsten。两者都运行相同的 BC 协议，但 Ropsten 中的加密货币没有实际价值。专用网络也是可能的。

与比特币相比，ETH 的关键创新是智能合约（SC）的支持。这些不是一些正式的要求或义务，但可以更充分地解释为自主代理人，其行为取决于他们的合同代码。每次此帐户收到消息时都会执行此代码，该消息是发送给它的消息。为了开发智能合约并因此提供去中心化应用程序（DApps），提供计算上图灵完备的（即 Turing complete）语言。基本 SC 语言是低级字节码语言，以太坊网络提供执行这种代码的虚拟机（即以太坊虚拟机，EVM）。几种高级（或更高）级别的语言可用于应用程序开发。目前的旗舰产品是 Solidity [19] - 一种类似 JavaScript 的语言，但过去曾使用其他语言。在以太坊虚拟机中执行之前，将较高级别的代码编译为字节码。

由于这些原因，以太坊成为（i）金融（货币，代币系统），（ii）半金融平台（例如人群感知）和（iii）非财务应用（在线投票，分散治理）。就创新方面而言以太坊在的区块链协议中是领先的。不列颠哥伦比亚省的大部分项目都旨在超越简单的价值交易，货币产品则以以太坊为基础。 以太坊的加密数字货币-ether-拥有第二大市值，仅比特币在其之前。

2.1.2. 超级账本 Fabric

超级账本项目是一项协作工作，用于创建企业级开源分布式账本框架和代码库。目前超级账本项目已成为 Linux 基金会的一个项目，目前拥有 130 多名成员，其中包括金融，银行，物联网，供应链，制造和技术领域的领导者。

超级账本 Fabric（HLF）[12]是目前在超级账本项目下孵化的多个项目之一，是一个主要针对商业用途的授权区块链平台。它是开源的，基于标准，运行任意智能合约（称为链式代码），支持强大的安全性，身份特征，基本的 REST API，CLI 以及使用可插拔共识协议的模块化体系结构（目前实现了拜占庭容错共识使用 PBFT 协议被支持）。该架构的分布式账本协议由同行运行。该结构区分两种对等体：（i）验证对等体是负责运行共识，验证事务并维护分类帐的网络上的节点，以及（ii）非验证对等体，其是充当代理的节点连接到验证同行[20]。

HLF 协议在例如 IBM Watson IoT™平台[21]。这是一个区块链及服务平台（BaaS），它使物联网设备能够将数据发送到私有区块链分类帐，以便将其包含在具有防篡改记录的共享交易中。 IBM Watson IoT 中的 HLF 主要适用于企业环境中的私有区块链，因为它使用与如以太坊之类的区块链平台采用不同的共识算法。 它的区别在于所有与区块链相关的功能都有详细记录的 HTTPS REST API。 Web 开发人员因此可以从区块链功能中受益，但继续使用 API 技术，他们已经熟悉了块和事务以及对等和网络，监视链状态以及区块链用户的注册和管理。

2.1.3. IOTA 区块链

IOTA [13]的发展是由 IOTA 基金会于 2015 年发起的。与 BC 技术相比，IOTA 分布式账本的主要特点是最初设计用于应对 IoT 带来的挑战。这些包括可扩展性（IOTA 分类账不限制交易的数量或评级），零交易费用（不同的共识原则）和快速的交易确认。以这种方式向物联网提供的新选项包括机器到机器纳米支付，安全（传感器）数据馈送和物联网设备识别。

与以太坊或比特币不同，以太币或比特币在一定程度上被作为加密货币的巨大成功所困，并尝试通过附加和扩展协议（状态通道，替代共识机制的考虑）来应对物联网的挑战，IOTA 是基于新的无块分布式账本架构。这是用有向无环图（DAG）实现的，称为纠缠。它提供的交易验证方法与以太坊和比特币中的块的挖掘完全不同。对于每个已发布的事务，IOTA 节点必须批准来自缠结中的其他节点的两个事务，而且不收费。与今天的区块链项目相反，共识不再是解耦的，而是系统的固有部分，导致分散和自我调节的对等网络。 IOTA 的关键限制之一是 DAG 缺乏交易订单。这实际上限制了 IOTA 智能合同的可能性。

IOTA 应用程序环境由 IOTA 节点组成。它们由核心或光源（用于资源受限的边缘设备）客户端与纠纷通信以及应用程序部分组成。客户端的参考实现是 Java，但其他的都包含在开发路线图中[22]。一个具有完整 API 覆盖的 JavaScript 库可用于 NodeJS 和浏览器应用程序。

所有 API 调用都将通过 JSON 格式的 POST HTTP 请求发送。客户端提供核心 IOTA 功能，可以通过 IOTA eXtension 接口（IXI）扩展模块。 IOTA 模块之一是用于安全，加密和授权数据流服务的掩码验证消息（MAM）。

公有区块链网络 IOTA 网络中提供可交换的加密货币。

3.物联网区块链应用程序开发

分布式账本为交易交易提供了一个值得信赖的环境。 就物联网的应用开发而言，两种模式可以组合在一起，即前端和上链。 根据预期用途，两种应用部件可以组合成一种解决方案。 前端应用程序部件是 Web，移动和嵌入式应用程序，它们通过 BC 客户端公开的客户端 API 使用 BC。 用户界面和物联网设备需要使用前端应用部分。 链上业务逻辑是指智能合约（即 HLF 中的链式代码），它们是在 BC 网络中部署和执行的程序。 卑诗省对智能合同的执行进行了验证。 BC 因此为智能合同执行提供了分散且可信的虚拟机。 物联网并非绝对需要链上逻辑。

3.1 基于区块链的应用

分布式环境的可信交易消除了对可信中央当局的需求。例如，基础适用于加密数字货币。然而，一些区块链技术技术超越提供智能合约 – 区块链的真正革命性特征，而不存在于传统的 Web，云，混搭体系结构中。智能合约是在区块链网络内执行的链式业务逻辑。执行可以由任何网络参与者验证，因此可以像 BC 网络中的任何其他事务一样进行信任。

智能合同代码被编写成相应的编程语言（例如，在用于以太坊的 Solidity 中，在用于吃啊记账本的 Go（或 Java）中），它被编译成适合于特定区块链的位代码，并被部署到网络。

一旦部署在区块链网络中，智能合同就按其唯一地址进行寻址，类似于常规的区块链帐户。智能合约暴露了功能，可以被其他区块链账户使用。这些功能代表其他区块链帐户的链式 API，可通过区块链访问。智能合约接收发往其的交易，参数中嵌入交易中的特定功能所需的参数。智能合约根据其编程逻辑处理传入请求并可选择启动事件。

3.1.1 前端应用部分

Web，移动或嵌入式应用程序将常规应用程序逻辑（例如用于操作用户界面，获取传感器数据，本地数据处理）与区块链功能相结合。后者可以是区块链网络中的简单交易交换或与链上应用部分（即智能合约）的通信。前端应用程序通过区块链客户端公开的区块链客户端 API 库和区块链客户端 API 使用区块链。前面的应用程序部分的这些功能块在本节的后续部分中有更详细的描述。

用户界面和物联网设备需要前端应用部分来利用 BC。前端应用程序可以在启用 BC 的 Web 浏览器/钱包中运行，例如， Mist [23]或带有 Metamask 浏览器插件的 Chrome 和 Firefox [24]。这使得基于 Web 的用户界面（UI）的高效实现成为可能。当前端部分需要 UI 并且开发人员想要依赖已知的 Web UI 技术（例如 HTML5，JS）时，浏览器被选为应用程序执行环境，但仍然在其解决方案中应用 BC。除了常见的浏览器功能（HTTP / HTTPs 协议，HTML 呈现）之外，启用区块链的 Web 浏览器还实现了应用程序使用区块链所需的区块链客户端 API 库，以及用于安全存储和管理用户区块链的关键钱包账户。启用区块链的 Web 浏览器是最简单的，但当然不是实现 UI 的唯一选择。高级用户界面可以建立，例如作为跨平台桌面应用程序，但开发人员需要导入用于区块链通信的库并自行实现关键钱包。

无人操作的嵌入式物联网系统在没有直接用户干预的情况下运行，因此浏览器不适合执行应用程序。在这种情况下，应用程序通常在某些服务端运行时环境中执行（例如，针对 JS 的 NodeJS），并且需要在该环境中导入适当的 BC 客户端 API 库才能正常运行。这是具有区块链支持的物联网设备的基础。对于启用 BC 的物联网设备来说，有两种关键操作模式可以与 BC 的变化一起工作并对其做出反应。在第一种情况下，设备由 BC 地址/帐户标识。 区块链交易可以发往和来自这个地址。对于要由发行人正确签署的传出交易，需要安全访问账户密钥库的位置（详见第 4 节）。在这种模式下，设备/应用可以例如自主记录其在链中的状态。在第二种情况下，物联网设备没有自己的区块链帐户。但即使没有它，设备也可以拦截 BC 网络中的智能合约创建的交易或事件。以这种方式，如果相应的事务或事件被记录在链中（例如，将某个值的事务处理到指定的 BC 地址），则应用程序可以执行某些动作（例如在继电器上切换）。这种操作模式是被动的，物联网设备/应用程序不能创建事务（作为嗅探器），但在安全密钥库管理方面要简单得多。尽管它们的范围相当不同，但这两种操作模式对于前端应用都具有实际价值。

前端应用程序中有四个关键功能模块，用于提供所需的功能并与区块链进行适当的通信：

•区块链客户端负责运行区块链协议，从而负责与 BC 网络的整个通信。这包括块管理（保持本地链最新）和交易（例如发送传出交易），监听事件，对等和网络管理，监控链状态，管理账户或挖掘块。有几个 ETH 区块链客户端实现可用，但 geth [25]通常作为参考，因为它由以太坊底层 Foundation 开发人员开发。一个普遍的选择是平价[26]。在 IOTA 中，区块链客户端是 IOTA 核心[27]。

•BC 客户端 API 是公开客户端功能的 BC 客户端的一部分。通过这个客户端 API，可以利用 BC 客户端的全部功能。可以通过公共编程和通信接口（通常是进程间通信（IPC），HTTP POST 或 Websockets（WS））来访问 API。如果应用程序和 BC 客户端在同一台机器上运行（本地通信），则可以应用 IPC，另一方面，HTTP 和 WS 也可以启用对 BC 客户端的远程访问。通过这些渠道之一传递的数据通常被构造为 JSON。

BC 客户端 API 库便于应用程序开发和 BC 客户端 API 的使用。这些库有各种实现，可用于不同的编程语言和不同的开发人员。在以太坊中，这样一个库是 web3.js [28]（当前版本为 0.20.x，已经公布了 1.0.0 测试版），在 IOTA 中是 iota.lib.js [29] - 全部用于 JavaScript。其他实现的成熟度可能会有所不同。这些库包含在应用程序项目中。除了连接 BC 客户端

API 之外，这些库还可以提供其他功能。一个是本地钱包，用于保存和保护对用户帐户和密钥的访问，并允许签署即将离任的交易。例如，此功能在以太坊 web3.js 0.x.x 中不受支持，但在 1.0.0 测试版中添加。这对 IoT 区块链应用程序开发至关重要，因为现在应用程序代码可以轻松，安全地管理帐户，无需用户交互。专用 ETH 钱包/浏览器需要用户确认新的传出交易，这不适用于嵌入式设备和应用程序。 区块链客户端 API 库提供用于签署事务并将其传递到 BC 网络的功能。在前端应用程序的编程语言中，事务是作为数据结构呈现的，然后传递给相应的函数。该数据结构没有签名字段，但通常会定义源地址。例如，在 JS 编程中使用 web3.js 1.0.0 库，函数 sendTransaction（）以 JSON 格式接收这样的结构，创建适当的签名，在 RLP 中编码结果以构建原始事务并将其广播到 BC 网络对等体。另一方面，signTransaction（）只创建一个原始事务，然后可以通过例如以后的方式传递给网络。

sendSignedTransaction（）。要对事务进行签名，sendTransaction（）和 signTransaction（）需要访问未锁定的 ETH 帐户。

•应用程序实现所需的功能，并通过区块链客户端 API 库使用区块链。应用程序编程代码是以太坊和 IOTA 经常用 JavaScript 编写的。原因是双重的。首先，在这两种情况下，JS BC 客户端 API 库都是最先进和经过验证的，其次，它适用于基于物联网设备应用程序的浏览器。

4 以太坊基于物联网设备的前端应用架构概述

前端应用部分的体系结构在很大程度上取决于部署前端的物联网设备的功能和局限性。物联网设备展示了从哑传感器节点到设备齐全的计算机等广泛的通信（比特率，连接持续性）和计算（CPU，存储）功能。因此，有必要事先了解这些功能，以便正确选择特定功能块（第 3.2 节）可以运行的位置以及它们的配置方式。例如，以太坊客户端可以运行各种区块链同步选项（完整的客户端 - 整个链数据（块头和块体）被下载并存储在设备中;轻客户端 - 仅获得当前链状态，更快同步，但是，不可靠的事件过滤）。

所有关于体系结构和配置的注意事项旨在为前端应用程序逻辑和与特定设备相关的以太坊事务（创建，签名，提交和监视）提供可靠的执行。

如何设计支持以太坊的物联网设备的前端应用架构有多种选择。这些选项在 IoT 设备的计算，通信和安全要求上有所不同。计算和通信限制主要与链式数据的同步（巨额）有关。如果在设备中应用非常低的比特率信道（例如低功率广域网（LPWAN）），通信也可能成为交易交换的问题。架构方面的安全性是指 JSON-RPC 的事务签名和 HTTP 和 WS 通道访问控制所需的密钥存储区的位置和访问。

在支持 ETH 的物联网设备中，前端应用架构的选项如下：

•具有 geth 客户端和应用程序部分的独立节点运行在同一物理设备和本地密钥存储区中。

•远程 geth 客户端，其中 JS 应用程序和 BC 客户端 API 库在受约束的物联网设备上运行，但 BC 客户端（geth）在另一个功能更强大的计算机/服务器上运行。在这种情况下，存在两

个子选项：

•物联网设备中的本地密钥存储区或远程密钥存储在运行 geth 的服务器上。

4.1.1 独立的物联网节点架构

在图 2a 所示的独立节点体系结构中，所有功能块都运行在同一物理设备中。 由于 BC 客户端（geth）也在那里运行，它对 CPU 和内存提出了很高的要求。 如果启用完整的 BC 同步，则必须计数几 GB 的 ETH 链数据才能传输并存储在设备中。

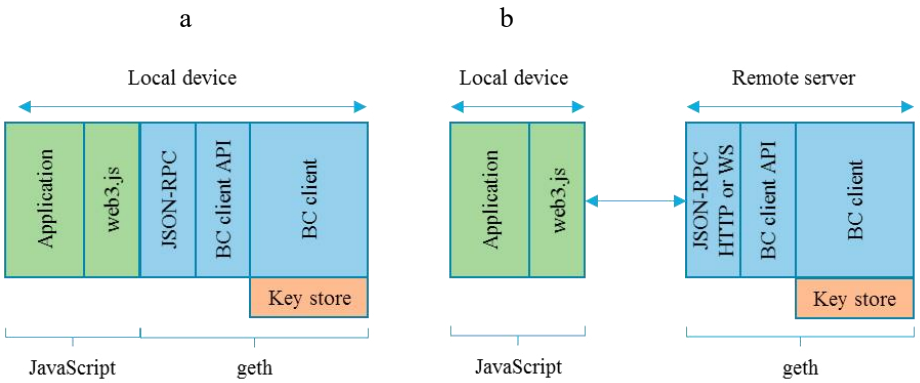

图 1.（a）独立的 IoT 节点和（b）远程 geth 客户端体系结构

这种情况下的密钥存储放置在本地，并在 BC 客户端初始化时通过 geth 解锁。这种情况下的主要风险是硬件安全性（如果物理设备隐私被侵犯，则被盗密钥）。

我们使用这种设置的经验表明，它仅适用于最强大的（IoT）设备。我们试图通过有线互联网连接在 RPi v3B 嵌入式系统上运行完整的客户端。该链的同步被证明是非常不可靠的。我们经历了异常的长时间同步（同步持续了几天，但仍未完成），意外的同步中断等。在进行这些测试时，我们有一个在普通计算机上运行的参考客户端（相同的 IP 网络容量）并且同步没有问题。重要的是要知道，未同步的 BC 阻止应用程序部分使用 BC 服务。我们也尝试在轻量级模式下运行 geth。同步更为成功，但我们在过滤由我们的智能合约启动的事件时遇到严重问题。由于所提供的数据信息不完整，有些事件丢失了，尽管相应的事务被记录下来并且链接同步。

4.2 远程 geth 客户端与远程密钥存储

通过远程 geth 客户端和远程密钥存储，我们在单独的无约束服务器上运行 geth。JavaScript 应用程序当然仍然保留在本地 IoT 设备中。通过这种方式，最需要资源的部分将

从物联网设备中移出。远程服务器通过 JSON-RPC API 公开 Geth 功能，HTTP 或 WS 作为传输通道。在这种设置中，密钥存储保留在服务器上，并在 geth 初始化时应用，就像独立节点一样。图 1b 描述了具有远程密钥存储体系结构的远程 geth 客户机。

这种架构实际上证明具有实际价值。本地设备成功运行应用程序部分，而远程服务器无缝运行 geth。我们使用 Wireshark 分析了本地设备和远程服务器之间的流量。一个典型的事务由应用程序以 JSON-RPC 形式通过 HTTP 提交给 geth，由一个 HTTP POST 请求和响应组成。请求消息的大小大约为 800B，JSON 格式的典型事务约为 450B。其余的消息是 TCP / IP 协议头。响应消息较小，为 280B。如果使用 WS 代替 HTTP，则消息大约小 200B。这似乎并不多，但仍可能超过低比特率设备的通信限制。尤其如此，如果不仅有限数量的事务通过 HTTP / WS 传递，而且还应用了一些事件过滤形式 web3.js，它利用轮询原则，生成恒定的网络负载。

但是，这种架构存在我们需要了解的潜在安全风险。如果 geth 在解锁密钥存储区的情况下运行，那么使用 HTTP 或 WS 上的 JSON-RPC 访问 geth 的任何人都可以创建使用此密钥签名的交易。对 geth 中建立的 HTTP 或 WS 没有访问控制，所以我们需要在这种情况下非常小心地规划 IP 网络层的安全性。只有在物联网设备充当主动交易创建者的情况下，这些风险才有意义。如果它运行在被动模式下（嗅探交易和事件链），则不需要密钥存储，因此在这方面没有风险。

4.3 具有本地密钥存储的远程 geth 客户端

具有本地密钥存储体系结构的远程 geth 客户机和具有远程密钥存储体的远程 geth 客户机在密钥存储的位置上有所不同。由于这不再放在服务器上，几个设备共享同一个 geth 服务器的安全风险就会减少。然而，在这种情况下，应用程序必须（创建和）提交原始交易，包括签名并应用适当的序列化。为了做到这一点，需要新的/额外的 JavaScript 库。 web3.js 版本 0.x.x 不支持签名事务。在这种情况下，必须应用轻钱包或 Ethereumjs-tx 等附加库。幸运的是，web3.js 版本 1.0.0.beta 宣布了保留和管理本地钱包所需的所有功能。我们认为这是实施具有本地密钥存储体系结构的远程 geth 客户端最可取的方法。

有趣的是，通过 HTTP / WS 传递原始事务而不是 JSON 事务对象并不会导致较小的消息大小，这是由于 RLP 编码效率更高所致。原始事务包括签名和事务散列（不存在于 JSON 事务对象中），在这种情况下导致大约 40B 的较大消息。

4.4 专有的本地设备到远程服务器通信

作为最后一种选择，我们看到 IoT 本地设备之间的专有通信协议结束了远程（geth）服务器，同时也丢弃了现有的 JSON 或 RLP 数据格式。我们看到它有两个好处。首先，通信

带宽要求可以降到最低，从而也能够通过低比特率通道进行通信。其次，我们可以应用高级服务器访问控制，以最小化远程密钥存储体系结构中远程 geth 客户端中描述的安全风险。

这种体系结构在一定程度上脱离了分布式对等理念，这是分布式分类帐和区块链的基础。另一方面，采取类似的方法，例如，在大多数移动 BC 客户端（移动应用必须信任提供 BC 功能的服务器）。最近的雾计算发展和 4 / 5G 网络架构也表明，网络边缘节点可以作为应用网关，为终端节点提供这种功能。

5.结论

通过我们的研究，我们期望澄清如何将物联网设备的要求和约束与适当的架构方法相匹配，以开发用于以太坊的前端物联网应用。 我们目前正在开发工具来自动生成和监控物联网设备中的交易，这将对上述体系结构进行系统性能测试。 我们将继续进行网络负载测量和流量分析，对通过低功耗，低比特率移动技术连接的以太坊物联网节点进行仿真和仿真研究。

致谢

作者感谢斯洛文尼亚研究机构资助的研究计划“电信算法和优化程序”的支持。

参考文献

[1] Dechamps A, Duda A, Skarmeta A, Lathouwer BD, Agostinho C, Cosgrove-Sacks C, et al. Internet of things applications - from research and innovation to market deployment. [Internet]. Vermesan O, Friess P, editors. Delft: River Publishers; 2014. (River Publishers Series in Communications). Available from: http://www.internet-of-things-research.eu/pdf/IoT-From%20Research%20and%20Innovation%20to%20Market%20Deployment_IERC_Cluster_eBook_978-87-93102-95-8_P.pdf

[2] Kasey Panetta. Gartner’s Top 10 Strategic Technology Trends for 2017 - Smarter With Gartner [Internet]. 2016 [cited 2018 Jan 25]. Available from: http://www.gartner.com/smarterwithgartner/gartners-top-10-technology-trends-2017/

[3] Mulholland A. Blockchain or Distributed Ledger? Defining the requirement, not the technology [Internet]. Constellation Research Inc. 2017 [cited 2018 Jan 25]. Available from: https://www.constellationr.com/blog-news/blockchain-or-distributed-ledger-definingrequirement-not-technology-0

[4] Share&Charge - Charging Station Network - Become part of the Community! [Internet]. [cited 2018 Jan 25]. Available from: https://shareandcharge.com/en/

[5] Brainbot Technologies AG [Internet]. Smart Contract and Blockchain Consulting for Enterprises. [cited 2018 Jan 25]. Available from:

http://www.brainbot.com/

[6] Ford N. IoT Application Using Watson IoT & IBM Blockchain [Internet]. Mendix. 2017 [cited 2018 Jan 25]. Available from: https://www.mendix.com/blog/built-iot-application-10-days-using-watson-iot-ibm-blockchain/

[7] Bocek T, Rodrigues BB, Strasser T, Stiller B. Blockchains everywhere - a use-case of blockchains in the pharma supply-chain. In:
2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). 2017. p. 772–7.

[8] Zerado Access Control [Internet]. Zerado. [cited 2018 Jan 25]. Available from: http://zerado.com/en/products-services/access-control/ [9] Huh S, Cho S, Kim S. Managing IoT devices using blockchain platform. In: 2017 19th International Conference on Advanced Communication Technology (ICACT). 2017. p. 464–7.

[10] IBM Watson Internet of Things. Blockchain and IoT: Vending Machine with eSIM Demo [Internet]. 2017 [cited 2018 Jan 25]. Available from: https://www.youtube.com/watch?v=T9kYuBcOnjI

[11] Viktor Trón, Felix Lange. Ethereum Specification [Internet]. 2015 [cited 2018 Jan 25]. Available from: https://github.com/ethereum/go-ethereum/wiki/Ethereum-Specification

[12] IBM Blockchain based on Hyperledger Fabric from the Linux Foundation [Internet]. 2017 [cited 2018 Jan 25]. Available from:
https://www.ibm.com/blockchain/hyperledger.html

[13] IOTA Developer Hub [Internet]. [cited 2018 Jan 25]. Available from: https://dev.iota.org/ [14] Protocol documentation - Bitcoin Wiki [Internet]. [cited 2018 Jan 25]. Available from: https://en.bitcoin.it/wiki/Protocol_documentation

[15] The Raiden Network [Internet]. High speed asset transfers for Ethereum. 2016 [cited 2018 Jan 25]. Available from: http://raiden.network/

[16] Lightning Network [Internet]. Scalable, Instant Bitcoin/Blockchain Transactions. [cited 2017 May 5]. Available from: http://lightning.network/

[17] Oraclize Documentation [Internet]. Overview. [cited 2018 Jan 25]. Available from: http://docs.oraclize.it/#overview

[18] Gavin Wood. The “Yellow Paper”: Ethereum’s formal specification [Internet]. 2017 [cited 2017 Aug 6]. Available from:
https://ethereum.github.io/yellowpaper/paper.pdf

[19] Solidity — Solidity 0.4.19 documentation [Internet]. [cited 2018 Jan 25]. Available from: http://solidity.readthedocs.io/en/latest/index.html

[20] Cachin C. Architecture of the Hyperledger Blockchain Fabric. In: Workshop on Distributed Cryptocurrencies and Consensus Ledgers
[Internet]. Chicago, Illinois, USA: ACM; 2016 [cited 2017 Jun 21]. Available from:
https://www.zurich.ibm.com/dccl/#program [21] IBM Watson Internet of Things. Blockchain and IoT: Vending Machine with eSIM Demo [Internet]. 2017 [cited 2018 Jan 25]. Available from: https://www.youtube.com/watch?v=T9kYuBcOnjI

[22] D. Sonstebo, “IOTA Development Roadmap,” IOTA, 31-Mar-2017. [Online]. Available: https://blog.iota.org/iota-developmentroadmap-74741f37ed01. [Accessed: 02-Aug-2017].

[23] Ethereum/mist: Mist [Internet]. Browse and use ?apps on the Ethereum network. [cited 2018 Jan 25]. Available from: https://github.com/ethereum/mist/#mist-browser

[24] MetaMask [Internet]. Brings Ethereum to your browser. [cited 2018 Jan 25]. Available from: https://metamask.io/

[25] Viktor Trón, Felix Lange. Geth [Internet]. ethereum/go-ethereum Wiki · GitHub. 2017 [cited 2018 Jan 25]. Available from:
https://github.com/ethereum/go-ethereum/wiki/geth

[26] Parity Ethereum Browser [Internet]. Parity Technologies. [cited 2018 Jan 25]. Available from: https://parity.io/

[27] IOTA Reference Implementation - iri [Internet]. GitHub. [cited 2018 Jan 25]. Available from:
https://github.com/iotaledger/iri

[28] web3.js - Ethereum JavaScript API [Internet]. GitHub. [cited 2018 Jan 25]. Available from:
https://github.com/ethereum/web3.js

[29] iota.lib.js - IOTA Javascript Library [Internet]. GitHub. [cited 2018 Jan 25]. Available from:
https://github.com/iotaledger/iota.lib.js