arXiv:2301.06210v2 [cs.DC] 4 Apr 2023

V-Guard：一种在动态成员关系下实现共识的高效许可型区块链

Gengrui Zhang、Yunhao Mao、Shiquan Zhang、Shashank Motepalli 和 Hans-Arno Jacobsen多伦多大学

摘要

本文介绍了 V-Guard，这是一种新型许可型区块链，可在动态成员关系下针对车辆数据实现共识，旨在解决车联网中车辆在道路上经常间歇性连接的问题。为实现这一目标，V-Guard 将成员管理整合到数据条目达成共识的过程中。它将数据条目与成员关系配置文件绑定，该文件描述了负责就该数据条目达成共识的车辆。因此，V-Guard 会生成数据条目及其对应成员关系配置文件的链式共识结果，从而使系统能够在动态成员关系下无缝达成共识。此外，V-Guard 将交易排序与共识分离，允许并发排序实例和周期性共识实例对数据条目进行排序和提交。这些特性使 V-Guard 能够在动态成员关系下高效实现共识，并具备高吞吐量和低延迟。

1 引言

自动化技术在现代车辆中的广泛应用，使汽车制造商成为集中式的数据垄断者。这些制造商收集消费者车辆产生的大量数据，但用户往往无法访问这些数据，必须获得制造商许可才能访问 [9, 91, 95]。车辆运行数据缺乏透明性和控制权，这日益引发人们的担忧；尤其是随着有关尚不成熟的自动化技术发生故障的报道和调查不断增多 [4, 5, 20, 23, 96]。例如，当事故相关数据只有一个来源——数据垄断者——时，很难确定驾驶员应承担多大责任 [8, 93]。不足为奇的是，近期调查显示，消费者认为数据完整性和透明性是他们对自动化技术最关切的问题 [1, 64]。此外，保险公司和立法者也对汽车行业的数据管理实践感到担忧。保险公司需要车辆运行和安全性数据，以改进其风险模型和定价 [55, 107] ，而立法者则致力于确保可问责性和安全性 [8, 92]。

此外，近年来，保障消费者的数据所有权已成为许多法规关注的重点。例如，欧盟《通用数据保护条例》（GDPR）与《加利福尼亚州隐私权法案》（CPRA）在数据管理方面秉持相似原则，包括数据完整性、透明性、所有权和可问责性 [99]。因此，要求提高车辆数据 [50, 114] 透明性并进行独立验证的呼声日益高涨。

1.1 为什么需要区块链?

目前使用的中心化解决方案，例如制造商数据采集器和事件数据记录器，往往无法遵循 GDPR 和 CPRA 所阐述的透明数据管理与数据所有权保护原则。首先，由于依赖单一事实来源（制造商），它们容易受到操纵。其次，用户必须获得制造商的许可，才能访问自己的数据。第三，对额外硬件设备的需求可能会使这些解决方案的成本效益降低 [43, 67]。

相比之下，区块链解决方案凭借其去中心化特性，能够从根本上打破数据垄断，消除来自单一“事实”来源的操纵可能性。它们可以强化数据所有权保护和透明性原则 [28, 48]。具体而言，车联网（V2X）中的区块链解决方案可以规避上述缺点。车联网允许车辆通过无线通信技术进行通信，例如专用短程通信（DSRC）和蜂窝车联网（C-V2X）。因此，数据可以通过联合共识以不可篡改的方式记录下来，从而防止用户和制造商伪造或删除证据。由于车辆在制造时已具备身份 [3], ，因此可以采用能够审查参与者身份的许可型区块链，以高性能实现确定性共识 [65, 80]。此外，将区块链解决方案作为独立软件部署在车辆的操作系统上，与使用事件数据记录器相比，可以降低产品和维护成本。这些特性不仅让用户能够更好地控制自己的数据，还能促进该行业的可问责性与安全性 [18]。

1.2 为什么需要一种新的区块链?

宝马 [19],梅赛德斯 [15,49], 大众汽车 [115, 116], 丰田 [72], 和福特 [11, 51]等领先汽车制造商已启动区块链项目。然而，这些应用采用传统的许可型区块链（例如 HyperLedger Fabric [7],CCF [102],和

Diem [37]），主要专注于供应链管理、网约车和汽车共享 [30, 31, 66]。这些区块链的共识算法（例如实用拜占庭容错 [25] 和 HotStuff [119]）专为在数据量相对较小的稳定环境中运行而设计。网络中的每台服务器都配置了预定义的成员关系配置，其中通常包含一组静态服务器。当发生成员变更（例如新服务器加入或旧服务器离开）时，共识算法会暂停正在进行的服务，并使用额外的重配置方法（例如 MC [101]和 Lazarus [52]）更新每台服务器上的成员关系配置，以一致地管理成员变更 [75, 83]。

这种暂停并更新的方法会在事务共识过程中造成"硬性中断"，当成员变更不频繁时或许可以接受。然而，对于车联网中使用的区块链而言，这种方法并不适用，因为由车辆任意连接状态引发的频繁成员变更的动态环境十分常见 [6, 53,98]。例如，在 DSRC车联网中，行驶车辆附近的车辆会在道路上动态变化，并根据距离上线或离线。同样，在蜂窝车联网中，很难预测哪些驾驶员会关闭车辆电源，导致参与者无法使用。

为应对频繁的成员变更，传统的拜占庭容错共识算法（例如实用拜占庭容错 [25] 和 HotStuff [119]）会遭受显著的性能下降 [42, 82]。例如，MC [101] 重配置需要约 10- 20 秒才能完成一次成员变更。在系统工作负载较高、规模较大时，成员管理带来的额外开销会变得十分显著 [2, 16, 101]。因此，亟需为车联网区块链提出新的解决方案，使其能够在成员变更期间无缝实现共识，同时不妨碍正在进行的共识过程。

1.3 V-Guard 对动态成员关系的支持

为应对车联网面临的独特挑战，我们提出了 V-Guard，这是一种采用新型共识算法的许可型区块链架构。V-Guard 旨在解决成员关系动态变化时如何高效实现共识这一挑战。与仅专注于数据交易共识的传统拜占庭容错算法（例如实用拜占庭容错 [25] 和 HotStuff [119]）不同，V-Guard 将成员资格配置协议纳入数据交易的共识中。它为数据交易及其对应的成员配置提供可追溯且不可变的账本。因此，V-Guard 的共识目标变为：

<数据交易、成员配置>

V-Guard 使用成员管理单元（MMU）处理动态成员配置。MMU 会记录所有可充当潜在成员的可用车辆，并将它们分成不同的组。它会生成一个成员配置文件队列，称为 booth（成员配置组），并为共识服务提供接口。在共识过程中，数据条目会在排序实例和共识实例中分别与指定的 booth 关联。当当前 booth 不可用时，排序／共识实例会从 MMU 获取一个新的 booth。因此，V-Guard 不仅就提交哪些数据达成共识，还就由哪个 booth（车辆）提交数据达成共识。

此外，由于不同传感器生成的车辆数据通常彼此没有交易关联，V-Guard 通过将排序与共识分离，支持对数据进行并发排序；排序阶段和共识阶段可以在不同的 booth（成员配置组）中进行。排序阶段将数据条目并发追加到全序日志中，而共识阶段则定期将已排序的条目提交到不可变账本中。这种分离最大限度地减少了消息传递，使 V-Guard 能够实现高吞吐量。此外，这种分离使 V-Guard 能够以更精细的方式管理成员资格；它使数据条目能够在不同的 booth（成员配置组）中进行排序和提交；也就是说，即使在动态成员关系下，也能无缝实现共识。

我们实现了 V-Guard，并评估了其在静态和动态成员关系下的性能。V-Guard 实现了高共识吞吐量和低延迟。例如，当成员数量为 $n = 4$时，在静态成员关系下，V-Guard 的峰值共识吞吐量达到 765,930 TPS（每秒事务数），延迟为 143 毫秒；在动态成员关系下，峰值共识吞吐量达到 615,328 TPS，延迟为 1324 毫秒。V-Guard 旨在仅记录和复制反映自动驾驶系统状态与决策的关键数据，因为需要记录的数据相对较少。这些数据包括速度、加速度、方向和目标检测等必要变量，而非原始数据。因此，所实现的性能表明，V-Guard 是一种切实可行的解决方案。

总而言之，V-Guard 具有以下创新特性。

- 它能够在成员变更期间无缝实现共识，且不会妨碍事务共识。它将成员管理整合到共识过程的各个阶段，旨在解决车联网区块链中频繁发生成员变更的问题。

- 它将排序与共识分离，从而支持并发数据排序，降低排序延迟和消息开销。这种分离还使数据条目能够在不同的成员资格中分别进行排序（排序）和提交（共识）。
- 其实现无论在静态还是动态成员关系下，都具有出色的吞吐量和延迟性能。该实现为开源项目，可在以下地址获取：https://github.com/vguardbc/vguardbft。

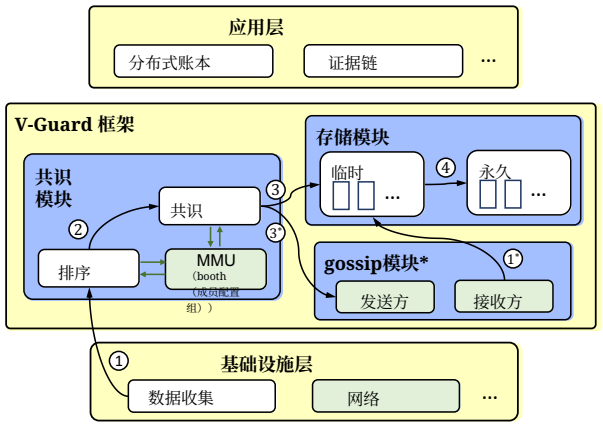


图 1. V-Guard 架构。共识模块以排序和共识实例的形式运行，并与 MMU 交互。已提交的数据存储在存储模块中，并通过 gossip 模块进一步传播。

2 V-Guard 概述

V-Guard 为车辆及其制造商提供区块链服务，以避免数据由单一方控制。要使用此服务，车辆会创建一个V-Guard实例，并以提议者的身份开始运行。车辆也可以作为验证者加入其他车辆的实例。每个 V-Guard实例必须至少包含四名成员：提议者、汽车制造商（又称枢轴验证者）以及其他车辆（至少两辆，又称车辆验证者）。以下介绍各角色。

提议者（V_p）。每辆车都是其自身 V-Guard 实例的提议者，因此每个实例只有一个提议者。提议者在其 V-Guard 实例中担任领导者：它向验证者提出数据，并协调验证者对所提议数据达成共识。**验证者**。在一个 V-Guard 实例中，除提议者外，其他成员均担任验证者。汽车制造商始终担任**枢轴验证者**（记作 V_π），其他车辆则担任**车辆验证者**（记作 V_i）。

一辆车可以同时参与多个 V-Guard 实例：它既是自身 V-Guard 实例的提议者，也是其他车辆的 V-Guard 实例中的验证者。我们将车辆加入的实例数量称为参与因子，γ。

参与因子（γ）。车辆的参与因子（$\gamma \in \mathbb{Z}^+$）描述了该车辆当前参与的 V-Guard 实例数量。当车辆加入或退出其他车辆的 V-Guard 实例时，参与因子也会相应变化。

V-Guard作为一个中间件系统运行，独立于其所在的汽车操作系统。它接收来自车辆基础设施层的输入，用于数据收集（例如来自传感器和监控设备），与其他车辆通信，并提供区块链服务，为车联网中的分布式应用提供支持。一般而言，如图1所示，一个V-Guard实例由共识和存储两个核心模块以及一个可选的泛洪传播模块组成（该模块并非正确性所必需）。

- 共识模块负责为收集到的数据条目协调共识（§4）。提议者使用其成员资格管理单元（MMU）获取有效的成员资格（§4.1）。随后，它启动一个排序实例，与所获取配置中的其他验证者协调，并为数据条目获取唯一的排序标识符（§4.2）。当按照各自成员资格配置定期调度的共识实例提交已排序的条目时，即达成共识（§4.3）。

- gossip模块可插拔，不会影响系统正确性（§5）。它旨在通过进一步向网络中的更多车辆传播并复制已提交的数据条目，提高稳健性。

- 存储模块用于存储共识模块中已提交的数据条目（§6）。默认情况下，它根据预定义策略（例如，24 小时）临时存储所有条目，并提供选项供用户或应用程序永久存储选定数据。

3 系统模型

V-Guard 是一种许可型区块链，需要成员提供身份信息；也就是说，每辆车都可以识别与其通信的对象。由于车辆在制造时便拥有身份信息[1]，，这种许可型区块链的标准做法在实践中切实可行。

服务属性。 V-Guard 的区块链架构提供基于证据的保障，确保车辆的行为符合预期。例如，它可用于记录应用自动化技术所产生的决策数据。**V-Guard 提供两项保证：任何一方都不能 ① 伪造证据，也不能 ② 掩埋证据**。 数据条目由独立车辆组成的法定人数进行排序和提交，并附有这些车辆的签名。达成共识后，数据条目可追溯且不可篡改。此外，提议者和共识枢纽都不能以数据丢失为由掩埋证据，因为每个已提交的数据条目都会一致地存储在参与共识过程的车辆中。

网络假设。V-Guard 不依赖网络同步性来保证安全性，但要求部分同步以确保活性；也就是说，在未知的全局稳定时间（GST）之后，两辆车辆之间的消息传递时间和消息处理时间均有上界。网络可能无法传递消息、延迟消息、复制消息或乱序传递消息。

[1]车辆识别号码（VIN） [3] 如同车辆的指纹，因为在用车辆的 VIN 各不相同。

容错。V-Guard 假设采用拜占庭故障模型，其中故障成员的行为可能是任意的，但成员故障彼此独立。每个成员都是独立实体，因为无论底层操作系统如何，V-Guard 都会独立运行；也就是说，当 f 车辆（无论其制造商是谁）发生故障（单独发生或相互串通）时，均计为 f 次故障。请注意，车辆参与多个 V-Guard 实例（$\gamma > 1$）时，其行为可能并不一致；也就是说，它可能在一个实例中正常运行，却在另一个实例中表现异常。

此外，我们不考虑故障成员发起的垃圾流量攻击（例如 DDoS攻击），因为仅靠共识算法无法处理此类行为。通常可以在更低层处理这些攻击，例如采用速率控制和黑名单方法。

密码学原语。V-Guard采用（t， n）门限签名，其中 t 个 n 成员可以共同对消息 [79, 106]进行签名。门限签名可以将 t 部分签名的消息（大小为 O（n））转换为一条完整签名的消息（大小为 O（1））。随后，所有 n 成员都可以验证这条完整签名的消息，从而证明 t 个成员已对其签名。利用门限签名实现线性性，已成为当前先进线性拜占庭容错协议中的常见做法，例如 HotStuff [119]，、SBFT [58], 和 Prosecutor [121]。

关于变更成员的说明。与其他拜占庭容错协议类似，V-Guard 要求总共 $n = 3f + 1$ 个成员中至少有 $2f + 1$ 个成员构成法定人数，才能容忍 f 个拜占庭故障。由于 V-Guard 旨在为车辆及其制造商提供证据链，因此在成员变更期间，我们始终将车辆的制造商（即枢轴验证者）纳入成员之中；因此，新展位最多可有 $n - 2$ 个新成员，提议者和枢轴验证者均可见证每个数据条目达成一致。

4 V-Guard 共识协议

本节将详细介绍 V-Guard 共识协议。如图3所示，V-Guard 有两类服务：事件驱动和守护进程服务。事件驱动服务会在收集数据时调用。通过编排这些服务，可对收集到的数据达成共识，并形成一条记录数据和成员信息的区块链。守护进程服务根据车辆的可用情况，动态建立和断开与车辆的网络连接，从而处理网络连接。

4.1 成员管理单元（MMU）

MMU 会跟踪网络中可用的车辆连接，并准备一个有效 booth（成员配置组）队列；动画示例可在 V-Guard 的 GitHub 页面上找到 [12])。展位（记作 $\mathcal{V}$）是一个成员配置文件，用于描述一组参与成员（车辆）。该配置文件包含车辆信息列表，包括其网络地址（即 v.conn）和公钥（即 v.pub）。展位的结构如下所示，

```
var boothQueue =struct { sync.RWMutex b []Booth //可用展位队列 } type Booth struct{ v []struct{ conn net.Conn //连接信息 pub share.PubPoly //个体公钥 ... } thpub share.PubPoly //门限签名公钥 ... }
```

每个 booth（成员配置组）的规模是在系统设置中预先设定的参数。有效的 booth 必须包含提议者（V_p）、枢轴验证者（V_π）以及其他作为车辆验证者的车辆。由于 V-Guard 假设存在拜占庭容错故障，因此 booth 的最小规模为四个（$3f$+1），以容忍 f=1 个拜占庭故障 [47]。

MMU 接口为排序和共识实例提供可用的 booth（分别记为 $\mathcal{V}_o$ 和 $\mathcal{V}_c$）。如果队列中没有可用的 booth（例如，车辆在隧道中没有网络连接时），排序和共识实例就会等待接口提供新的 booth。此外，MMU 会根据 booth 的可用性和效率动态管理 booth。当 f 辆车辆不可用时，它会移除相应的 booth，并定期 ping 其他车辆，优先选择网络延迟最低的 booth。

在每个 booth 中，提议者首先为（t， n）门限签名建立分布式密钥生成（DKG），其中 t 是法定人数规模， n 是 booth 规模。DKG 完成后，同一 booth 中的所有车辆共享同一个用于门限签名的公钥（thpub）。需要指出的是，虽然 booth 外的车辆无法验证该门限签名的公钥，但仍可验证单个车辆的公钥（即 v.pub），因为所有车辆都使用相同的 V-Guard 密钥生成服务。

借助 MMU 的特性，V-Guard 能够在动态成员关系下执行排序和共识流程。例如，图 2 展示了一个包含 10 辆参与车辆的 V-Guard实例，其中 V_p 是提议者， V_π 是枢轴验证者，而 V_0 至 V_7 是 8 个车辆验证者。在此情况下，提议者为数据条目 1 发起排序实例时，该流程在展位1（初始展位）中进行。下一个排序实例（即数据条目 2）若仍可使用当前展位（展位1），便会继续使用。当当前展位不可用时（例如，部分车辆离线），MMU 将为下一个排序实例（即数据条目 3）提供一个新展位（展位2）。因此，每个排序实例都会生成一组成对结果：一笔已排序数据交易及其所在的成员信息（详见 §4.2）。预定共识实例执行时，会与

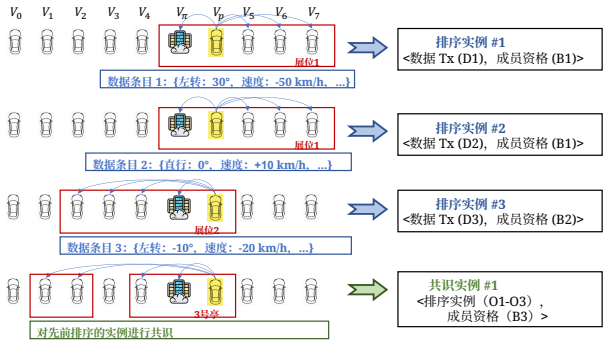


图2. MMU 嵌入式排序和共识过程示例。当当前 booth（成员配置组）不再适用于排序或共识实例时，提议者（V_p）会从 MMU 获取一个新的 booth（成员配置组）。成员资格配置的一致与每个排序和共识实例中的数据交易的一致相配对。

MMU 交互，采用相同的机制复用当前展位或获取新展位。共识实例通过提交成对结果，维持排序实例的输出，从而确保这些结果在不同成员资格下均不可变。借助 MMU 与排序/共识实例的协同机制，V-Guard 能够在成员变更时无缝实现共识，并生成不可变账本，记录可追溯的数据交易及其对应的成员关系配置。

4.2 排序——形成全序日志

排序阶段为全序日志中的每个数据批次分配唯一的序列号（顺序）。V-Guard 将数据批次的排序与共识分开。这种分离实现了非阻塞排序过程：数据批次的排序无需等待其前一个批次的共识完成。这种分离还使排序和共识能够在成员变更期间于不同的 booth（成员配置组）中进行。这两项特性使 V-Guard 更适用于车辆通过不稳定网络连接的车联网，也提高了其效率。

排序阶段实例会将日志条目追加到全序日志中，以表明 ① 达成一致的数据是什么，② 顺序如何，以及由 ③ 哪些车辆成员组成。具体而言，首先，当收集到的数据达到提议者预先设定的批次大小（记为 β）时，提议者便会启动一个排序实例。随后，排序实例会将提议者收集的数据批次分发给所有验证者，包括枢轴节点和车辆（"达成一致的数据"）；提议者会为每个数据批次分配唯一的排序标识符（即序列号）（"顺序"），并指定成员资格配置，以表明哪些验证者作为法定人数对该排序标识符达成一致（"成员资格"）。下面介绍数据批次的排序阶段实例的工作流程。

O1 提议者 V_p 为一个数据批次（记为 $\mathcal{B}_i$）分配唯一的序列号（记为 i）作为排序标识符（记为 O_i），并启动一个排序实例。

a. V_p 向 MMU 确认此排序实例（$\mathcal{V}_o$）的展位是否可用且为最新；如果不是，则调用 MMU 获取新的 $\mathcal{V}_o$。b. V_p向 $\mathcal{V}_o$ 中的所有成员发送 预排序 $\mathcal{B}_i$, $h_{\mathcal{B}_i}$, $\mathcal{V}_o$, $h_{\mathcal{V}_o}$, i, σ_{V_P}，其中 $h_{\mathcal{B}_i}$是 $\mathcal{B}_i$的哈希值；$h_{\mathcal{V}_o}$是 $\mathcal{V}_o$的哈希值；σ_{V_P} 是 V_p对 i、$h_{\mathcal{B}_i}$和 $h_{\mathcal{V}_o}$的组合所做的签名。c. V_p 创建集合 $\mathcal{R}_{o_i}$，等待来自 $\mathcal{V}_o$ 中验证节点的回复。

O2 收到预排序消息后，V_i 首先验证消息，然后回复 V_p。当 ① h_B 和 $h_{\mathcal{V}_o}$ 分别与 B和 $\mathcal{V}_o$的哈希值匹配、② σ_{V_p} 有效且 ③ i 尚未使用时，验证成功。如果验证成功，V_i 则向 V_p发送 PO-Reply i，σ_{V_i}，其中 σ_{V_i}是V_i对 i、h_B和 $h_{\mathcal{V}_o}$的组合所做的签名。

O3 V_p 从验证者处收集预排序回复消息，并将其添加到 $\mathcal{R}_{o_i}$ 中，直到收到 $2f$条回复（即 $|\mathcal{R}_{o_i}| = 2f$）；然后，V_p 执行以下操作。

a. V_p 将 $2f$收集到的签名（即 σ_{V_i} 在 $\mathcal{R}_{o_i}$）转换为一个（t, n）门限签名 σ_{o_i}，其中 $t = 2f$.
b. V_p 创建一个子集 $\mathcal{Q}_{o_i}$，其中包括 $2f$车辆，其签名会从 $\mathcal{V}_o$ 转换为 σ_{o_i}，作为成员法定人数；即 $|\mathcal{Q}_{o_i}| = 2f \wedge \mathcal{Q}_{o_i} \subset \mathcal{V}_o$。
c. V_p向 $\mathcal{V}_o$中的所有成员发送 Order i，$\mathcal{Q}_{o_i}$，σ_{o_i}。

V_p 现在认为批次 $\mathcal{B}_i$ 已按排序标识符 O_i排序（$O_i = i$），该批次由来自 $\mathcal{V}_o$的成员中的法定人数 $\mathcal{Q}_{o_i}$ 确认。它将 O_i，$\mathcal{B}_i$，$\mathcal{Q}_{o_i}$，$\mathcal{V}_o$，σ_{o_i} 追加到全序日志中。

O4 收到 Order 消息后，V_i 会根据三项标准对其进行验证：① σ_{o_i} 有效，且阈值为 $2f$（法定人数规模正确）；σ_{o_i} 的签署者 ② 与 $\mathcal{Q}_{o_i}$ 匹配（有效验证者）；并且③ $\forall V_i \in \mathcal{Q}_{o_i}$，$V_i \in \mathcal{V}_o$（成员资格相同）。如果验证成功，$V_i$ 会认为批次 $\mathcal{B}_i$ 已由 $\mathcal{V}_o$中的法定人数 $\mathcal{Q}_{o_i}$按 ID O_i排序。随后，它将 O_i，$\mathcal{B}_i$，$\mathcal{Q}_{o_i}$，$\mathcal{V}_o$，σ_{o_i} 追加到全序日志中。

在每个排序实例中，每条数据记录都会与一个可验证委员会配对。验证者会接收委员会信息（$\mathcal{V}_o$），并在 O1 中根据收到的委员会验证法定人数；在 O4中，配对信息会使用唯一的排序标识符进行排序，并在一个共识实例中提交。

4.3 共识——提交全序日志

共识阶段确保展位中的所有正确成员对全序日志上显示的条目达成一致，并生成一个定期提交这些日志条目的不可变账本。

穿梭共识。V-Guard 会定期启动共识实例，以提交追加到全序日志中的日志条目。每个共识实例类似于一辆"穿梭

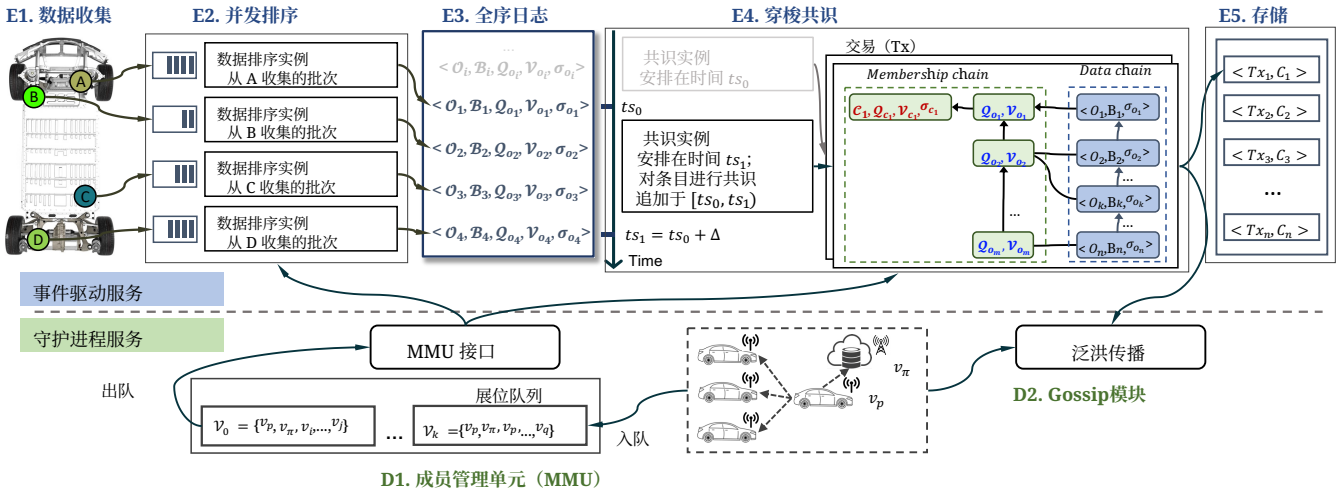


图3。 V-Guard 工作流程。MMU 为排序实例和共识实例准备 booth（成员配置组）队列。排序实例并发地将数据条目追加到全序日志中。共识实例将已排序的条目聚合为交易，并定期提交。已提交的交易可以通过泛洪传播进一步传播给其他车辆。

巴士”，按固定时间间隔启动，间隔记为 Δ（例如 $\Delta = 100$ 毫秒），并负责处理在 Δ 期间追加的日志条目。形式上，在时间 $ts+\Delta$ 安排的共识实例负责处理在 $[ts, ts+\Delta)$ 期间追加的日志条目，其中最初 $ts = 0$。因此，共识实例涵盖全序日志中的所有条目；也就是说，**任意两个相邻的共识实例之间都不会遗留未提交的日志条目**。我们使用起始时间戳（即 $[ts, ts+\Delta$ 中的 ts）作为相应共识实例的 ID（即 $\mathcal{C}_i = ts$）。

成员关系剪枝。 我们将共识实例中所有待提交的条目 $\mathcal{C}_i$ 称为一笔交易（tx）；即 $tx = \{entry_{ts}, \ldots\ldots, entry_{ts+\Delta}\}$。一笔交易可以包含多个日志条目，这些条目的排序是在相同的成员资格下进行的。在这种情况下，交易会剪除冗余的成员关系和法定人数信息，将在同一展位中排序的条目关联到相同的成员关系配置。例如，在图 3 E4 中，如果条目 $\{\langle \mathcal{O}_2, \mathcal{B}_2 \rangle \ldots\ldots \langle \mathcal{O}_k, \mathcal{B}_k \rangle\}$由成员资格配置 $\langle \mathcal{Q}_{O_2}, \mathcal{V}_{O_2}, \rangle$排序，那么它们只会关联到一个成员关系配置。当成员资格很少变化时，剪枝可以在保留成员关系可追溯性的同时减小消息大小。应用剪枝后，具有 ID $\mathcal{C}_i = ts$（即安排在时间 $ts+\Delta$）的共识实例按如下方式运行。

C1 V_p 为追加到 $[ts, ts+\Delta$ 中的日志条目准备一笔剪枝后的交易（tx）。

a. V_p 向 MMU 确认此共识实例（$\mathcal{V}_c$）对应的 booth（成员配置组）是否可用且为最新状态。请注意，如果 $\mathcal{V}_c = \mathcal{V}_o$，则 $\mathcal{V}_o$ 仍然可用；否则，$\mathcal{V}_c$ 是 MMU 提供的新 booth（成员配置组）。b. V_p向 $\mathcal{V}_c$中的所有 V_i发送预提交消息。由于共识阶段可能在不同的 booth（成员配置组）中进行（即 $\mathcal{V}_c \neq \mathcal{V}_o$），$V_p$会检查 V_i之前是否在法定人数中，并且是否已看到 tx中的条目。

- 如果 V_i已看到 tx中的所有条目，V_p会发送 tx中条目的起始和结束排序 ID（O_{ts}和 $O_{ts+\Delta}$），以便 V_i能从其日志中找到 tx。V_p发送

预提交 ts，h_{tx}，O_{ts}，$O_{ts+\Delta}$，$\mathcal{V}_c$，$h_{\mathcal{V}_c}$，σ_{V_p}，其中，h_{tx}和 $h_{\mathcal{V}_c}$分别是 tx和 $\mathcal{V}_c$的哈希值，而 σ_{V_p}是 V_p对 ts、h_{tx}和 $h_{\mathcal{V}_c}$的组合进行签名所用的签名。

- 如果 V_i尚未见过 tx中的条目，V_p就会发送预提交 ts，h_{tx}，tx，$\mathcal{V}_c$，$h_{\mathcal{V}_c}$，$\{\mathcal{R}_{o_i}\}$，$\{\mathcal{Q}_{o_i}\}^{\sigma_{V_p}}$，并捎带待提交的条目（$tx$）；其中，$\{\mathcal{R}_{o_i}\}$是 tx中每个条目对应的签名回复集合，$\{\mathcal{Q}_{o_i}\}$是相应签署者的集合（从排序中的步骤 O3 获取）。

c. V_p 创建一个集合 $\mathcal{R}_c$，等待验证者的回复。

C2 收到预提交消息后，V_i 首先根据以下标准验证该消息。

- 如果 V_i已见过 tx，V_i则定位从 O_{ts}到 $O_{ts+\Delta}$ 的日志条目，并计算它们的哈希值。① 该哈希值必须等于 h_{tx}，② σ_{V_p}才有效，并且 ③ ts尚未被其他共识实例使用。

- 如果 V_i不在排序区块中（尚未看到 tx），那么 V_i就无法验证门限签名，因为 V_i未参与该区块中的 DKG 生成过程（见 §4.1）。因此，要验证前一个排序阶段的正确性，它必须验证$\{\mathcal{R}_{o_i}\}$中包含的每个条目的回复 tx。相应验证者的公钥可在$\{\mathcal{Q}_{o_i}\}$中找到。在这种情况下，① V_i会计算 tx的哈希值，且该哈希值必须等于 h_{tx}；②对于每个数据条目 $\in tx$，$2f+1$

签名回复可在 $\{\mathcal{R}_{o_i}\}$ 中找到，且这些签名符合 $\{\mathcal{Q}_{o_i}\}$；并且 ③ ts 未被之前的共识实例使用。

如果验证成功，V_i 会向 V_p 发送 PC-Reply $\mathcal{C}_i$，σ_{V_i}，其中 σ_{V_i} 是 V_i 对 ts、h_{tx} 和 $h_{\mathcal{V}_c}$ 的组合进行签名所得的签名。

C3 V_p 从验证者处收集 PC-Reply 消息，并将其添加到 $\mathcal{R}_c$ 中，直到收到 $2f$ 条回复（即 $|\mathcal{R}_c| = 2f$）；随后 V_p 执行以下操作。

a. V_p 将 $2f$ 收集到的签名（即 $\mathcal{R}_c$ 中的 σ_{V_i}）转换为一个（t，n）门限签名 $\sigma_{c_{ts}}$，其中 $t = 2f$。 b. V_p 从 $\mathcal{V}_c$ 中创建一个子集 $\mathcal{Q}_{c_{ts}}$，其中包括签名被转换为 $\sigma_{c_{ts}}$ 的 $2f$ 辆车辆，作为成员法定人数；即 $|\mathcal{Q}_{c_{ts}}| = 2f \wedge \mathcal{Q}_{c_{ts}} \subset \mathcal{V}_c$。 c. V_p 向 $\mathcal{V}_c$ 中的所有成员发送 Commit ts，$\mathcal{Q}_{o_i}$，$\mathcal{V}_c$，$\sigma_{c_{ts}}$。

V_p 现在将交易视为 tx 已提交，并将其广播 tx 给 gossip 模块中的已连接车辆（该模块将在 §5中介绍）。

C4 收到 Commit 消息后，V_i 会通过以下三项标准对其进行验证：① $\sigma_{c_{ts}}$ 有效，且达到阈值 $2f$；② 的签署者与 $\mathcal{Q}_{c_{ts}}$ 相符；③ $\forall V_i \in \mathcal{Q}_{c_{ts}}$，$V_i \in \mathcal{V}_c$。

如果验证成功，V_i 会将 tx 视为已提交。

通过采用相互独立的排序实例和共识实例，数据条目可以在不同的 booth（成员配置组）中进行排序和提交。然而，在每个排序/共识实例中，booth（成员配置组）不可更改（这是安全性所要求的）；也就是说，在 O1-O4 或 C1-C4期间，必须始终使用同一个 booth（成员配置组）。如果某个 booth（成员配置组）在实例完成前发生故障，该实例将中止当前排序/共识过程，并调用 MMU 获取新的 booth（成员配置组），随后重试相应的过程。

成员关系的一致性。 V-Guard的共识同时就数据条目和成员关系配置达成一致。如图 3所示，已提交的数据批次构成一条数据链，经裁剪的成员关系配置构成一条成员关系链。由于每个数据条目都与其所属实例的成员关系配置配对，因此成员关系与数据条目一并达成一致；二者均可通过参与成员的签名进行追溯和验证。V-Guard支持动态成员关系这一特性，使其适用于车联网，并能在其中高效运行。

此外，V-Guard 的消息复杂度为 $\mathcal{O}(n)$。排序阶段的步骤 O1 和 O2，以及共识阶段的 C1 和 C2，消息复杂度为 $\mathcal{O}(n)$，因为大小为 $\mathcal{O}(1)$ 的消息仅在提议者与 $(n-1)$ 个验证者之间传递。在步骤 O3 和 C3 中，提议者将 $2f$ 个签名（总大小为 $\mathcal{O}(n)$）转换为一个大小为 $\mathcal{O}(1)$ 的门限签名，并将其广播给 $(n-1)$ 个验证者，因此该步骤的消息复杂度仍为 $\mathcal{O}(n)$。因此，V-Guard 实现了线性消息复杂度。

4.4 正确性讨论

现在讨论 V-Guard 的正确性。与传统的拜占庭容错协议不同，V-Guard 不按视图依次运行，因此不采用视图变更。V-Guard 不依赖视图，而是使用动态成员管理系统来处理网络中的变化。此外，提议者同时承担客户端和领导者的角色，因为它是唯一产生数据的成员。当提议者发生故障时，其 V-Guard 实例会停止运行，而当其他车辆的 MMU 将其验证者纳入其中时，这些验证者将加入其他车辆的 V-Guard 实例。

定理 1（有效性）。 正确成员在共识阶段提交的每个数据条目，都必须已在排序阶段被提议。

证明。 只有车辆才能作为其发起的V-Guard实例的提议者。在O2中，验证者对与提议者相同的内容签名，其中包括所提议数据条目的哈希值。在O3 和O4中，有效的门限签名由法定人数成员对所提议数据条目的同一哈希值所作的签名转换而成。此外，提议者还会对包含该数据条目的交易的哈希值进行签名，该交易位于C1中。如果交易包含未被提议的数据条目，则C2 中的验证过程将失败。因此，已提交的交易必须包含在排序阶段已被提议的数据条目。 □

引理 1。 同一排序区块中的任意两个正确成员，对于同一数据条目的冲突排序 *ID*，不会达成一致。

证明。 在排序阶段 booth（成员配置组）保持不变时，这个引理显然成立，因为正确的验证者不会回复包含先前已分配排序标识符的 预排序 消息（③ 位于 O2中）。在动态 booth （成员配置组）下，由于 V-Guard 对法定人数构造的要求，每个法定人数都必须包含提议者和枢轴验证者；如果提议者将一个数据条目重复分配为两个不同的排序标识符，并将其发送到具有不同 booth（成员配置组）的两个排序实例，那么其中一个排序实例就会失败，因为两个 booth（成员配置组）都必须包含枢轴验证者，且该验证者不会与提议者串通。当枢轴验证者不作回复时，提议者无法继续执行 O3，该排序实例将被中止。因此，对于同一数据条目的冲突排序标识符，正确成员不会达成一致。 □

定理 2（安全性）。 在每个 *booth*（成员配置组）中的故障数少于 f 时，所有正确成员对提议的数据条目达成全序一致。

证明（概要）。我们通过反证法证明安全性定理。我们声称，两个正确成员按相同顺序提交这两个数据条目。也就是说，如果两个数据条目在具有相同排序的共识实例中提交，那么必定存在两个在 C3 中构造的法定人数，它们对这两个已追加到全序日志中的条目达成一致；

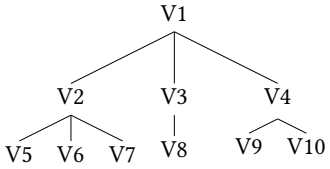


图4. 一条 Gossip 消息的传播树，包含 $\lambda = 2$。V_1 是提议者，其余为传播者。

否则，提议者无法在 C3 中获得足够的选票。在这种情况下，这两个条目必定在排序阶段被赋予相同的排序标识符，这与引理 1矛盾。因此，每个条目都以唯一的排序标识符提交。□

定理 3（活性）。 正确的提议者最终会在共识阶段收到对所提议数据条目的回复。

证明。 V-Guard 假设部分同步以保证活性。GST 之后，消息延迟和处理时间都有界。由于相邻的共识实例之间不会遗留未提交的日志条目，根据定理 1 和 2，在某个共识实例中未能提交的交易总会在后续的共识实例中得到处理。因此，在足够长的同步期间，正确的提议者最终会收到验证者的回复。 □

5 泛洪传播

泛洪传播模块通过进一步向网络传播已提交的交易来增强系统稳健性。该模块不会影响共识的正确性，可根据需要启用。启用泛洪传播后，共识模块提交一笔交易后，提议者会向其他已连接车辆发送一条携带该交易的 Gossip 消息，其他车辆则会继续将此消息传播给各自已连接的车辆。因此，消息传输形成一棵 传播树，其中提议者是名为 传播者 的根节点，其他节点则是泛洪传播节点。

每条Gossip 消息都有一个生存期（记为 λ），它决定消息在一次传输中经过的传播者数量，也就是传播树的高度。例如，在图4中，V_1 是提议者，并与 V_2、V_3和 V_4直接相连。该 Gossip 消息的生存期为2；当路径中包含2个传播者时，消息便停止传播；例如，路径 <V_1、V_2、V_5>。具体而言，我们将泛洪传播工作流程描述如下。

提议者： 提议者（V_p）通过执行以下操作，开始传播已提交的交易（即 C 中的提交消息3）。(a) V_p 为一条 $\lambda = 3$提交消息创建生存期（例如，$\lambda = 3$），并创建集合 $\mathcal{G}$ 以记录该消息传输路径的遍历信息。(b) V_p 对提交消息进行哈希，并对哈希结果（h_c）与 λ 的组合进行签名，以获得一个签名σ_{V_P}。随后，它添加一条遍历条目，其中包含 λ、σ_{V_P}和 C_{V_p}，并将其添加到 $\mathcal{G}$；即 $\mathcal{G}.add$（$[\lambda, \sigma_{V_p}, C_{V_p}]$），其中 C_{V_p}是 V_p的地址和公钥（§4.1）。

(c) V_p向未参与 Commit 消息共识过程的已连接车辆发送Gossip、Commit h_c，tx，$\mathcal{G}$，然后创建一个列表，用于接收来自传播者的确认消息。

传播者： 传播者（V_i）存储有效的 Gossip 消息；如果仍有剩余生存期，则可以进一步传播该消息。

(a) V_i 根据四项标准验证收到的 Gossip 消息： ① 之前未收到过此消息；② 中的签名有效； ③ λ中的 s 严格单调递减；以及 ④ 消息仍有剩余生存期；即， $\lambda_{min} = min\{\mathcal{G}.\lambda\} > 0$。如果验证成功， V_i 向该消息的提议者和枢轴验证者发送确认消息，并将自身登记到传播者列表中。

(b) V_i 递减此 Gossip 消息的生存期；即，λ_{new}= $\max\{\lambda_{min} - 1, 0\}$。如果 $\lambda_{new} > 0$，则 V_i准备进一步传播该消息。它对 h_c 和 λ_{new} 的组合进行签名，以获得签名 σ_{V_i}，并向 $\mathcal{G}$中添加一个新的遍历条目；即，$\mathcal{G}.add$（$[\lambda_{new}, \sigma_{V_i}, V_i]$）。

(c) V_i向其他已连接车辆发送 Gossip，Commit h_c，tx，$\mathcal{G}$，但不发送给那些向其发送此 Gossip 的车辆连接的车辆，不包括它通过此 Gossip#+#+#+#+#+ 消息的车辆（见 (a)）。

V-Guard 中的 gossip模块是一种额外机制，可提供额外一层冗余。由于每个传播者都向提议者注册，提议者可以连接到传播者并读取其自身部分的数据。当共识在较小的展位规模下进行时，这一点尤为重要，因为车辆故障可能会显著影响系统的整体韧性。尽管泛洪传播无法保证网络中的所有参与者都能收到所有交易，但即使面临网络分区或故障，它也有助于提高交易送达的概率，从而增强系统的稳健性，同时不干扰核心共识过程。

6 存储模块

V-Guard 将已提交数据批次存储在存储主机中。存储主机会为来自不同车辆的交易创建存储主实例（SMI），因为车辆可能在不同的 booth（成员配置组）中扮演不同角色（即 $\gamma > 1$）。例如，在图5中，存储主机有三个实例区域，分别包含提议者、验证者和传闻者 SMI（如果启用了泛洪传播模块）。车辆的存储主机最多有一个提议者 SMI，因为它是其自身 V-Guard实例的唯一提议者。当车辆加入其他 booth（以验证者身份运行）时，其存储主机会创建验证者 SMI。提议者和验证者 SMI 的数量等于车辆的参与因子（γ）。当启用泛洪传播时

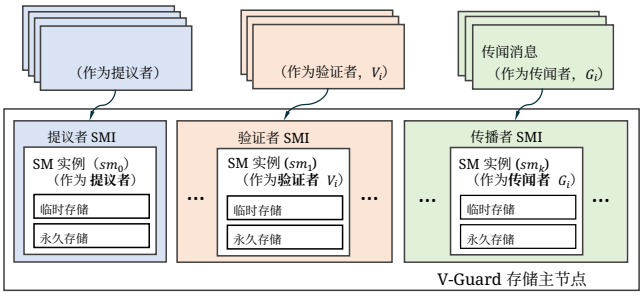


图 5. 存储主实例（SMI）。车辆在为自身执行共识时可运行提议者 SMI，在参与其他车辆的共识时可运行验证者 SMI，还可在启用 gossip 模块时运行传播者 SMI。

车辆可能会收到传闻消息，其存储主机会创建传闻者 SMI，以存储从其他车辆传播而来的共识结果。

每个存储主实例分为两层：临时存储和永久存储，旨在最大限度地利用车辆有限的存储空间。与运行在服务器上的区块链平台（例如，HyperLedger Fabric [7], CCF [102], 和 Diem [37]）不同，V-Guard运行于车辆之间，而车辆通常只有有限的存储能力。

临时存储根据预定义策略暂存交易。V-Guard的实现采用计时策略，定义存储主机保存一笔交易的时间段（记为 τ）（例如 $\tau = 24$ 小时）。默认情况下，交易会登记在临时存储中；如果用户没有发出后续指令，交易将在 τ 时间后删除。此外，用户可以要求将临时存储的交易转移到永久存储并永久保存。此选项将存储控制权交给用户。例如，当用户怀疑车辆发生故障时，他们可能希望保留相关交易作为证据，并将其移至永久存储。我们为这种分层设计介绍了四种存储主实例 API。

- sm.RegisterToTemp(&tx, time.Now())用于将一笔交易注册到临时存储层。
- sm.CleanUpTemp(time.Now())是一个守护进程，用于计算临时存储中交易的剩余时间，并删除已过期的交易。

- sm.MoveToPerm(&tx)由用户调用，用于将选定的交易从临时存储移动到永久存储。
- sm.DeletePerm(&tx)由用户调用。交易永久存储并达到用户的用途后，用户可以将其删除。

临时存储策略可以通过不同方式实现。例如，可以采用定义固定存储空间大小的策略。当临时存储超过预设大小时，它会按照先进先出（FIFO）的方式清除旧交易。

基于策略的存储模块可以显著减少存储空间的使用，因为车辆通常存储能力有限。请注意，枢轴验证者可以永久存储所有数据，因为汽车制造商通常在其云平台上使用可扩展的存储设备。

7 评估

我们将 V-Guard 实现（vg）的端到端性能与三种最先进的基准方案进行了比较（使用它们的开源实现）：Hot-Stuff [62, 119]（hs），一种线性 BFT 协议，其变体用于 Facebook Diem区块链 [37]；ResilientDB [60,100]（rdb），一种以 PBFT [25]为共识协议的 BFT键值存储；以及 Narwhal [35, 87]（nw），一种在共识前分发交易的基于 DAG 的内存池协议。我们在某主流云平台 [29]上的 4、16、31、61 和 100 个虚拟机实例上部署了这些方案。每个实例都包含一台机器，配备 2 个虚拟CPU，由 2.40 GHz 英特尔至强处理器（Skylake）提供支持，缓存大小为 16 MB，随机存取存储器为 7.5 GB，磁盘空间为 90 GB，运行 Ubuntu 18.04.1 LTS。通过 iperf 测得的 TCP/IP带宽，以及两个实例之间的原始网络延迟，分别约为 400 MB/s 和 2 毫秒。

我们引入以下符号来报告结果：

m 消息大小（字节）；即，为达成共识而发送的数据条目的大小，不包括任何消息头（消息中的其他参数）。β 批次大小；即，为达成共识而发送的数据批次中的数据条目（事务）数量。n 节点数（成员资格／集群规模）。δ 使用 netem 实现的模拟网络延迟。

网络延迟。我们使用 netem 在所有规模下模拟额外的网络延迟，延迟值服从正态分布，分别为 $\delta = 0$、10 ± 5、50 ± 10、100 ± 20 毫秒。模拟网络延迟下的性能变化反映了在更复杂网络（例如，无线网络）中的性能。

工作负载。V-Guard 是一种多功能区块链，允许用户定义自己的消息类型。消息大小是复制中的一个关键因素，因为所有消息都需要序列化后才能通过网络传输，无论其类型如何。为确保与其他基线方案进行公平比较，论文中的评估重点考察不同消息大小（$m = 32$、64 和 128 字节）下普通消息的复制性能。不过，值得注意的是，V-Guard 的消息服务经过设计，可轻松适配并支持不同类型的消息。一些项目已经利用这一特性开发了自己的消息类型，相关内容可在 V-Guard 的 GitHub 页面上找到 [12]。

本节其余部分安排如下。§ 7.1展示了将 V-Guard 与其在静态成员关系下的基线进行比较的评估结果；§ 7.2 报告了性能

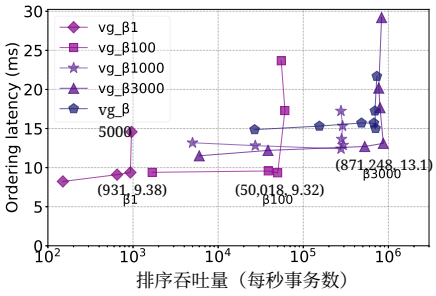


图 6。 不同批次大小下 V-Guard 的排序吞吐量与延迟，其中 $\beta = 3000$ 在 $n = 4$、$\delta = 0$ 和 $m = 32$ 条件下获得最佳性能。

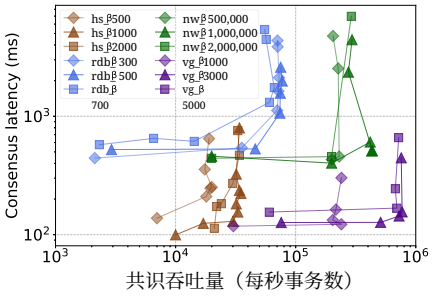


图 7。 在 $n = 4$、$\delta = 0$ 和 $m = 32$ 条件下，采用不同批次大小时 V-Guard 及其基线的共识吞吐量与延迟对比。

	批次大小	吞吐量（每秒事务数）	延迟 (ms)
HotStuff	(β=1000)	34，015	155
ResilientDB	(β=500)	80，580	4368
Narwhal	(β=10^{6*})	423，058	516
V-Guard (o)	(β=3000)	871，248	13
(c)		765，930	143

* Narwhal 使用固定大小的字节缓冲区作为批次，因此其缓冲区大小为 $= m \times \beta_b$。

图 8. V-Guard（包括排序 (o) 和共识 (c)）及其基线在各自最佳批次大小下的峰值性能汇总，其中$n = 4$、$\delta = 0$ 和 $m = 32$。

V-Guard 在动态成员资格下运行这一独特特性的性能；以及 § 7.3 总结了评估结果。

7.1 静态成员关系下的性能

我们将 V-Guard 的共识间隔时间设为 $\Delta = 100$ 毫秒；也就是说，每 100 毫秒调度一次共识实例。性能以吞吐量和延迟报告，其中吞吐量按每秒事务数（TPS）计算（即每种算法每秒提交的条目数），延迟则测量从发送请求到收到确认所经过的时间。

我们采用批处理来测量性能。随着批次大小不断增加，吞吐量的边际增幅逐渐减小，并在某个特定的（最佳）批次大小下达到峰值；此后，随着批处理在共识过程中变得比其他因素代价更高，吞吐量会下降。

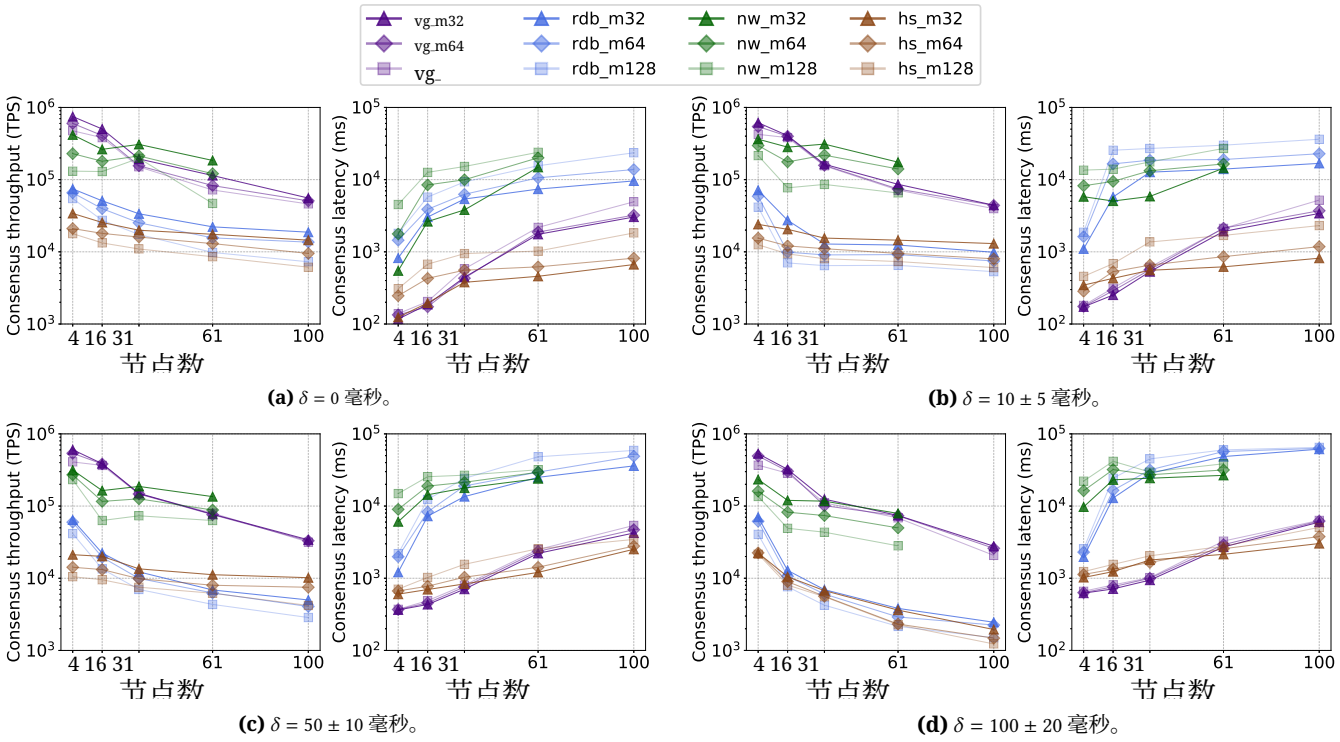
为进行公平比较，在与基线方案进行对比时，V-Guard 仅运行一个实例（即 $\gamma = 1$），见 §7.1.1 和 7.1.2；多个 V-Guard 实例同时运行时的性能见 §7.1.3。

7.1.1 峰值性能。所有方案的峰值性能均出现在 n=4 个节点、消息大小为 m=32 字节且未实现网络延迟（即$\delta = 0$ 毫秒）时。为了获得 V-Guard 的峰值性能，我们不断增大其批次大小并监测结果（如图6所示）：排序吞吐量起初不断提升，但边际收益递减（例如，$931 > {}^{50,018}_{100} = 500.18 > {}^{871,248}_{3000} = 290$），并在吞吐量达到871，248 每秒事务数、延迟为13 毫秒且 $\beta = 3000$时达到峰值。此后，在更高的 β 下，排序吞吐量下降而延迟上升。我们采用这种方法测量所有方案的共识吞吐量和延迟的峰值性能（如图7所示）。对于每种批次大小，我们不断增加客户端数量，直至网络达到饱和（即曲线出现拐点）。我们展示了 V-Guard 及其基线方案在三种批次大小下的结果，这些批次大小对应近乎最优的（例如，vg_1000）、最优（例如，vg_3000）和超最优性能（例如，vg_5000）。

图 8 总结了所有方法的峰值性能，V-Guard 在吞吐量和延迟方面均优于其基线方案。由于共识实例会定期提交有序数据条目，因此排序与共识具有相似的吞吐量。V-Guard 的高性能得益于其实现和设计。与 HotStuff [62] 和 ResilientDB [100], 的实现相比，V-Guard 使用门限签名，实现了线性消息复杂度。此外，分离式排序通过并发发起排序实例来降低延迟，而穿梭式共识则显著减少了消息传递，从而节省网络带宽并提高吞吐量。

7.1.2 可扩展性。随后，我们测量了在不同网络延迟下，随着消息大小和系统规模增加，V-Guard 及其基线方案的共识性能（见图9）。我们将每种方案的批次大小设为其在 n = 4条件下的最佳（最优）批次大小。结果表明，随着消息大小和网络延迟增加，所有方案的吞吐量均有所下降，延迟均有所上升；其中，消息大小对吞吐量的影响更为显著，而网络延迟则对共识延迟有更直接的影响。

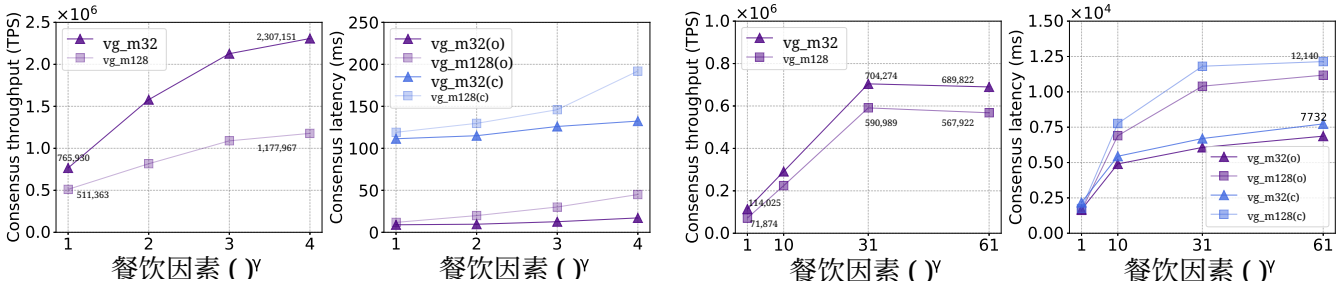
随着系统规模不断扩大，采用单一领导者管理共识和交易分发的 V-Guard、HotStuff 和 ResilientDB，吞吐量均有所下降。相比之下，当系统规模从 16 个节点扩大到 31 个节点时，Narwhal 的吞吐量有所提升。Narwhal 设计了一种内存池协议，将交易分发与共识分离。在 Narwhal 中，每个节点既作为工作节点，也作为验证者运行：工作节点通过基于 DAG 的内存池协议传播交易，而验证者则专注于就交易的哈希值达成共识。因此，当更多交易输入网络时，Narwhal 的吞吐量会随节点数从 16 增至 31 而提升；但当交易分发带来的负担超过共识过程时，吞吐量便开始下降



(a) $\delta = 0$ 毫秒。 **(b)** $\delta = 10 \pm 5$ 毫秒。

(c) $\delta = 50 \pm 10$ 毫秒。 **(d)** $\delta = 100 \pm 20$ 毫秒。

图 9. 在已实现的网络延迟条件下，当消息大小为$m = 32$、64和128 字节时，V-Guard 与其基线方案在各自最佳批次大小下的性能比较；正态分布网络延迟为 $\delta = 0$、 10 ± 5、 50 ± 10和 100 ± 20 毫秒，系统规模依次增大为 $n = 4$、16、31、61和100 个节点。（在 $n = 100$条件下，Narwhal 的内存池协议因所需内存空间超过系统资源而无法初始化。[2]）



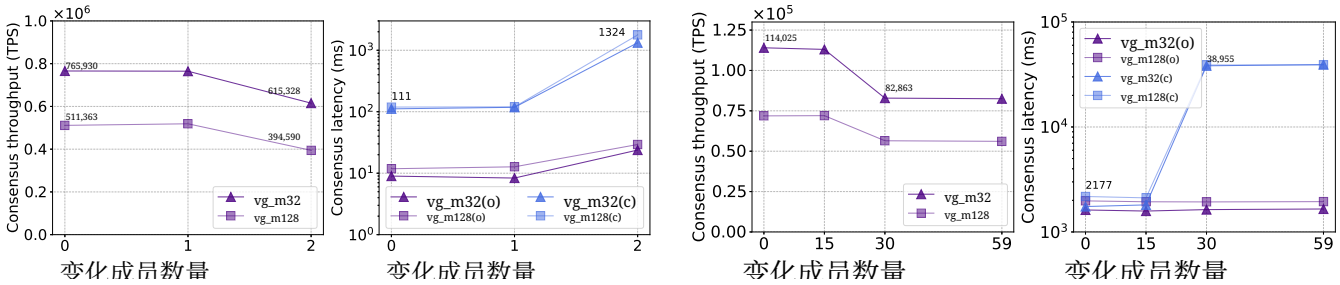
(a) $n = 4$。 **(b)** $n = 61$。

图10. V-Guard 在参与因子逐渐增加时的性能，其中 $\delta = 0$、 $m = 32$ 和 $\beta = 3000$。

过程。尽管 Narwhal 的吞吐量表现出色，但其有向无环图结构导致延迟较高，使 Narwhal 成为延迟最高的算法之一。

V-Guard 在小规模下实现了最高的吞吐量。这得益于排序与共识过程的分离，从而显著降低了消息传递开销。随着系统规模扩大，尽管 V-Guard 只有一个节点（提议者）负责传播数据条目，它仍然保持了与 Narwhal 相当的吞吐量，即使 Narwhal 使用有向无环图结构，并有多个节点传播数据条目。总体而言，V-Guard 将排序与共识过程分离，使其即使在大规模下也能以低延迟实现高吞吐量。

此外，在消息负载、系统规模和网络延迟增加时，与所有其他方案相比，V-Guard 的性能下降幅度最小。具体而言，与其基线方案相比，较高的网络延迟对 V-Guard 的共识吞吐量和延迟影响最小。例如，当 $n = 4$ and $m = 32$ 时，其吞吐量从网络延迟为 $\delta = 0$ 毫秒时的 765,930 每秒事务数降至网络延迟为 $\delta = 100 \pm 20$ 毫秒时的 531,594 每秒事务数，而延迟则从 143 毫秒升至 622 毫秒。



(a) $n = 4$。 **(b)** $n = 61$。

图11. V-Guard 在动态成员关系下的性能，其中 $\delta = 0$ 和 $\beta = 3000$。由于 V-Guard 的成员资格始终包含提议者和枢轴验证者，因此在 $n = 4$ 和 61 下允许的变更成员数分别为 2 个和 59 个节点。

V-Guard 的排序实例以非阻塞方式运行，无需等待先前实例完成；穿梭共识实例则将网络延迟分摊到 Δ 中包含的数据条目上。这一特性使 V-Guard 非常适合车联网应用，因为这类应用通常存在较高的网络延迟。

7.1.3 随参与因子增加时的性能。我们还评估了每个节点运行多个 V-Guard 实例（即 $\gamma > 1$）时 V-Guard 的性能。例如，在一个包含 4 个节点（n_1 至 n_4）且 $\gamma = 2$的系统中，n_1 和 n_2启动各自的 V-Guard 实例并担任提议者；n_1 也在 n_2的实例中担任验证者，反之亦然。其余两个节点 n_3 和 n_4 则在 n_1 和 n_2 的实例中都担任验证者。

我们测量了 V-Guard 在两种系统规模下，参与因子从 $\gamma = 1$增至 $\gamma = n$ 时的吞吐量和延迟（如图10所示）。当 n=4时，吞吐量随之增加，但增幅逐渐减小，峰值达到每秒 2,307,151 个事务，而排序和共识延迟均略有上升。当 n=61 且 $m = 32$时，吞吐量在 $\gamma = 31$ 达到峰值，为704,274 每秒事务数；在 $\gamma = 61$ 时，吞吐量降至 689,822 每秒事务数。参与因子越高，每个实例的性能就越低，因为系统资源需要由所有实例分摊。

7.2 动态成员关系下的性能

为展示 V-Guard 在成员资格频繁变更时实现共识的能力，我们评估了其在最坏情况下的性能：每个数据条目都在单独的 booth（成员配置组）中处理并提交。这意味着成员资格变更发生的速率与事务共识相同。

我们在两种规模下测量了动态成员关系下的性能：n=4 和 n=61。由于 V-Guard 始终将提议者和枢轴验证者纳入成员资格，因此在 n=4 和 n=61下，最多分别允许 2 个和 59 个成员（节点）发生变更。在这种情况下，每个排序和共识实例都会从 MMU 获得一个新的 booth（成员配置组）。例如，当 n=4 中有 2 个成员发生变更时，排序和共识 booth（成员配置组）有一个成员不同：

Ordering booth: [n_1 (as V_p), n_2 (as V_π), n_3, and n_4]
Consensus booth: [n_1 (as V_p), n_2 (as V_π), n_5, and n_6]

当 n=61时，我们评估了在 15、30 和 59 个变更成员的情况下，两种消息工作负载（m=32 和 128 字节）下的吞吐量和延迟性能。

结果表明，当变更成员数量少于 f时，动态成员关系不会影响性能。由于每个排序和共识实例都在不同的 booth（成员配置组）中进行，当某个共识实例包含此前未见过有序数据条目的新成员时，这些新成员必须执行一项耗时的任务，以验证交易中包含的所有数据条目签名。因此，旧成员比新成员更快地向提议者回复；当剩余旧成员仍能组成法定人数（$2f + 1$）时，变更成员不会影响系统性能；例如，图 11a 和 11b中分别有 1 个和 15个变更成员。

相反，如果共识法定人数中的剩余旧成员数量低于 $2f + 1$，法定人数就必须等待新成员加入。这会显著影响性能，因为新成员会成为瓶颈。由于新成员必须完成验证任务并响应提议者，共识延迟会增加，导致整体性能下降。当 $m = 32$时，在 $n = 4$ 和 $n = 61$下，吞吐量分别下降 19% 和 27%。在 $n = 61$中有 30 个新成员时，共识延迟从 2177 毫秒增加到 38,955 毫秒。

7.3 结果总结

在静态成员关系下，V-Guard 在吞吐量和延迟方面优于其基线方案（§7.1.1）。当系统

[2]Narwhal 的内存池协议要求根据系统规模和批次大小进行静态内存分配；当 n=100时，仅运行该程序之前，协议就要求每个工作节点为其分配 5.7 吉字节的静态内存，而我们的系统资源无法满足这一要求。

扩展规模时，V-Guard 在不同工作负载下的性能下降幅度与基线方案相近（§ 7.1.2）。随着网络延迟增加，在所有规模下，V-Guard 的吞吐量下降幅度均小于基线方案（§ 7.1.2）。随着 V-Guard 实例数量增加（$\gamma > 1$），吞吐量先上升至峰值，随后开始下降（§ 7.1.3）。在动态成员资格下，当新成员少于 f 时，V-Guard 的性能保持不变；当新成员超过 f 时，延迟会急剧增加，因为新加入的成员必须经过验证流程（§7.2）。

8 相关工作

车联网中的区块链。车联网区块链应能够实时就海量数据达成共识，例如由特斯拉自动驾驶仪 [112] 和凯迪拉克Super Cruise [22], 生成的车辆状态数据，并应能够应对道路上车辆之间的不稳定连接 [65, 80]。根据是否需要成员管理，区块链可分为无许可区块链和许可型区块链。无许可区块链采用Proof-of-X协议系列（例如工作量证明 [88] 和权益证明 [54]）来达成概率性一致，并允许参与者匿名加入。然而，由于性能有限，无许可区块链通常不会部署在需要实时处理海量数据的延迟关键型V2X区块链应用中 [45, 61, 68, 120]，尽管某些隐私关键型V2X用例要求匿名参与 [46, 94]。

相比之下，许可型区块链通过身份注册参与者，并凭借签名验证参与者，从而实现高吞吐量和低延迟。它们通常采用拜占庭容错共识算法（例如，实用拜占庭容错 [25],SBFT [58]和 HotStuff [119]），以实现所有参与者之间的确定性一致，并受到主要汽车制造商发起的早期区块链测试平台的青睐 [15, 19, 72, 115]。然而，这些区块链无法适应动态成员关系；成员变更时，共识必须暂时停止，以便更新配置文件 [101]。

BFT共识算法。共识算法是为状态机复制（SMR）提供安全性和活性的核心 [104]。拜占庭容错状态机复制提供了一种共识服务抽象，能够在存在拜占庭故障的情况下，就请求的全序达成一致 [10, 74, 76, 103]。随着参与用户可能故意违反协议（发起拜占庭攻击）以获取更多收益，区块链中的拜占庭故障正变得日益普遍 [34, 78, 122]。

基于领导者的拜占庭容错算法已被许可型区块链广泛采用，例如 HyperLedger Fabric [7]和 Diem [37]。在 PBFT [25] 率先提出实用的拜占庭容错方案后，人们从各个方面提出了大量优化方法。它们采用推测性决策，并减少单一领导者的工作负载 [40, 59, 71], 开发高性能实现 [17, 21, 44, 56,97, 109], 减少消息开销 [39, 81, 84, 89, 108, 118],通过使用阈值签名获得线性消息复杂度 [58, 119, 121], 将交易分发从共识中提取出来 [35],利用可信硬件限制故障行为 [13, 26, 36, 70, 77], 提高容错阈值 [63, 117], 通过秘密共享提供机密性保护 [113], 并对各个参与者实施可问责性 [27, 90, 105]。

除了基于领导者的算法，无领导者拜占庭容错算法可避免单点故障和单一领导者瓶颈 [33, 41, 73, 85, 111]。没有领导者时，无领导者拜占庭容错算法通常采用二元拜占庭一致协议 [86] 共同形成法定人数 [14]，但冲突解决的消息开销和时间成本较高。

成员重配置。成员变更涉及共识系统的重配置 [75]，其中一致协议需要切换到更新后的副本组配置 [38]。在弱一致性模型下，一些方法通过点对点查找，将某个标识符（例如， [24, 110]）的邻域确定为系统成员关系。在强一致性模型下，则需要一系列关于系统成员关系的一致视图，以确保共识过程的安全性 [32, 57, 69]。例如，MC [101] 提出了一种实用拜占庭容错成员重配置方法 [25]，，通过跟踪系统成员关系并定期通知其他系统节点成员变更。然而，事务共识必须等待每次成员重配置完成，因此在成员频繁变更时效率低下。

9 结论

V-Guard 是一种许可型区块链，能够在系统成员关系持续变化（动态环境）时高效实现共识。它将每个数据条目与系统成员关系配置绑定，并对提议的数据和成员关系配置分别获得链式共识结果。此外，它将排序与共识分离，使数据条目能够在不同成员之间进行排序和提交。这种分离在静态和动态环境中都带来了高性能。在静态成员关系下，V-Guard 的峰值吞吐量比具有最低共识延迟且性能最佳的基线高 1.8× 。在动态环境中，V-Guard 展现出在变更成员下运行的灵活性，其吞吐量达到静态环境下性能的 73%。

参考文献

[1] Hillary Abraham、Chaiwoo Lee、Samantha Brady、Craig Fitzgerald、Bruce Mehler、Bryan Reimer 和 Joseph F Coughlin。2016。《自动驾驶汽车、信任与驾驶替代方案：消费者偏好调查》。载于运输研究委员会第*96*届年会（2016）。

[2] Ittai Abraham、Dahlia Malkhi、Kartik Nayak、Ling Ren 和 Alexander Spiegelman。2016。《Solida：一种基于可重配置拜占庭共识的区块链协议》。arXiv 预印本 arXiv:1612.02916（2016）。[3] 美国国家公路交通安全管理局。 [无日期]。车辆识别号码（VIN）。 http://www.ibc.ca/qc/。[4] 美国国家公路交通安全管理局。2021。《总结报告：自动驾驶系统碰撞报告常设一般命令》。 https://www.nhtsa.gov/sites/nhtsa.gov/files/2022-06/ADS-SGO-Report-June-2022.pdf。[5] 美国国家公路交通安全管理局。2022。Waymo在美国国家公路交通安全管理局的自动驾驶汽车碰撞事故名单中位居榜首。https://www.freightwaves.com/news/waymo-tops-nhtsas-list-of-autonomous-vehicle-crashes/.[6] 5G Americas。2021。《车联网连接：蜂窝车联网与5G》。 https: // www.5gamericas.org/wp-content/uploads/2021/09/Vehicular-Connectivity-C-V2X-and-5G-InDesign-1.pdf。[7] Elli Androulaki、Artem Barger、Vita Bortnikov、Christian Cachin、Konstantinos Christidis、Angelo De Caro、David Enyeart、Christopher Ferris、Gennady Laventman、Yacov Manevich 等。2018。《Hyperledger Fabric：面向许可型区块链的分布式操作系统》。载于第十三届欧洲计算机系统会议论文集。ACM，30。[8] 美国律师协会。2018。《无人驾驶汽车的法律影响》。 https://www.americanbar.org/news/abanews/publications/yourabaa/2018/december-2018/legal-implications-of-driverless-cars/。[9] 汽车保养协会和汽车售后市场供应商协会（AASA）。2021。《你的汽车，你的数据，你的选择》。https://www.jobbernation.ca/your-car-your-data-your-choice-comes-to-canada/。[10] Hagit Attiya 和 Jennifer Welch。2004。《分布式计算：基础、模拟与进阶主题》。第19卷。John Wiley & Sons。[11] Ford Authority。2021。福特区块链倡议旨在识别来源不道德的材料。 https://fordauthority.com/2021/12/ford-blockchain-initiative-aims-to-identify-unethically-sourced-materials/.[12] V-Guard代码库。2023。V-Guard BFT。 https://github.com/vguardb c/vguardbft。[13] Johannes Behl、Tobias Distler 和 Rüdiger Kapitza。2017。《强化版混合方案：基于SGX的高性能拜占庭容错》。载于第十二届欧洲计算机系统会议论文集。222–237。[14] Michael Ben-Or、Boaz Kelmer 和 Tal Rabin。1994。《具有最优韧性的异步安全计算》。载于第十三届ACM分布式计算原理研讨会论文集。183–192。[15] 梅赛德斯-奔驰集团。2020。《请绕街区一圈！》https: // group.mercedes-benz.com/innovation/blockchain-2.html。[16] Alysson Bessani、Eduardo Alchieri、João Sousa、André Oliveira 和 Fernando Pedone。2020。《从拜占庭复制到区块链：共识只是起点》。载于2020年第50届IEEE/IFIP可靠系统与网络国际会议（DSN）。IEEE，424–436。[17] Alysson Bessani、João Sousa 和 Eduardo EP Alchieri。2014。《借助BFT-SMART实现面向大众的状态机复制》。载于2014年第44届IEEE/IFIP可靠系统与网络国际会议。IEEE，355–362。[18] 《利用区块链改进公共部门的数据管理》。2017。麦肯锡。 https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/using-blockchain-to-improve-data-management-in-the-public-sector。[19] 宝马。2021。《区块链汽车解决方案如何帮助驾驶员》。 https://www.bmw.com/en/innovation/blockchain-automotive.html。[20] 美国国家运输安全委员会。2021。《警方称，一辆无人驾驶特斯拉发生碰撞并燃烧了四个小时，造成得克萨斯州两名乘客死亡》。 https://www.washingtonpost.com/nation/2021/04/19/tesla-texas-driverless-crash/。[21] Ethan Buchman。2016。Tendermint：区块链时代的拜占庭容错。博士学位论文。[22] 通用汽车凯迪拉克。 [无日期]。凯迪拉克 Super Cruise。https://www.cadillac.com / world-of-cadillac/innovation/super-cruise。[23] CARSURANCE。2022年。24项自动驾驶汽车统计数据与事实。https: // carsurance.net/insights/self-driving-car-statistics/.[24] Miguel Castro、Peter Druschel、Ayalvadi Ganesh、Antony Rowstron 和 Dan S Wallach。2002。结构化点对点覆盖网络的安全路由。ACM SIGOPS《操作系统评论》36，SI（2002），299–314。[25] Miguel Castro、Barbara Liskov 等。1999。实用拜占庭容错。载于 OSDI，第99卷。173–186。[26] Byung-Gon Chun、Petros Maniatis、Scott Shenker 和 John Kubia- towicz。2007。经证明的只追加内存：让对手信守承诺。ACM SIGOPS《操作系统评论》41，6（2007），189–204。[27] Pierre Civit、Seth Gilbert 和 Vincent Gramoli。2021。Polygraph：可问责的拜占庭一致性。载于2021 IEEE第41届国际分布式计算系统会议（ICDCS）。IEEE，403–413。[28] James Clavin、Sisi Duan、Haibin Zhang、Vandana P Janeja、Karuna P Joshi、Yelena Yesha、Lucy C Erickson 和 Justin D Li。2020。政府区块链：用例与挑战。《数字政府：研究与实践》1，3（2020），1–21。[29] Compute Canada Cloud。 [无日期]。Arbutus云平台。https: // arbutus.cloud.computecanada.ca/。[30] CNBC。2018。全球四大汽车制造商希望将"区块链"引入你的汽车。 https://www.cnbc.com/2018/05/02/four-of-the-worlds-largest-automakers-want-to-bring-the-blockchain-to-your-car.html。[31] 福布斯技术委员会。2021。汽车行业区块链：三大用例与三大挑战。https://www.forbes.com/sites/forbestechcouncil/2021/12/22/blockchain-in-the-automotive-sector-three-use-cases-and-three-challenges/?sh=5dcd164f2508。[32] James Cowling、Dan RK Ports、Barbara Liskov、Raluca Ada Popa 和 Abhijeet Gaikwad。2009。Census：面向大规模分布式系统的位置感知成员管理。（2009）。[33] Tyler Crain、Vincent Gramoli、Mikel Larrea 和 Michel Raynal。2018。Dbft：高效的无领导者拜占庭共识及其在区块链中的应用。载于2018 IEEE第17届网络计算与应用国际研讨会（NCA）。IEEE，1–8。[34] Philip Daian、Steven Goldfeder、Tyler Kell、Yunqi Li、Xueyuan Zhao、Iddo Bentov、Lorenz Breidenbach 和 Ari Juels。2020。Flash boys 2.0：去中心化交易所中的抢先交易、矿工可提取价值与共识不稳定性。载于2020年IEEE安全与隐私研讨会（SP）。IEEE，910–927。[35] George Danezis、Lefteris Kokoris-Kogias、Alberto Sonnino 和 Alexander Spiegelman。2022。Narwhal and Tusk：基于DAG的内存池与高效BFT共识。载于第十七届欧洲计算机系统会议论文集。34–50。[36] Jérémie Decouchant、David Kozhaya、Vincent Rahli 和 Jiangshan Yu。2022。DAMYSUS：利用可信组件实现精简型BFT共识。载于第十七届欧洲计算机系统会议论文集。1–16。[37] Diem。2020。Diem区块链。https://developers.diem.com/main/docs/the-diem-blockchain-paper。[38] Tobias Distler。2021。从系统视角看拜占庭容错状态机复制。《ACM计算调查》（CSUR）54，1 2021 1 38( )，–。[39] Tobias Distler 和 Rüdiger Kapitza。2011。通过按需副本一致性提升拜占庭容错系统的性能。载于第六届计算机系统会议论文集。91–106。

[40] Sisi Duan、Sean Peisert 和 Karl N Levitt。2014。hBFT：以最低成本实现推测式拜占庭容错。IEEE《可靠与安全计算汇刊》12，1（2014），58–70。[41] Sisi Duan、Michael K Reiter 和 Haibin Zhang。2018。BEAT：让异步BFT变得实用。载于2018年ACM SIGSAC计算机与通信安全会议论文集。2028–2041。[42] Sisi Duan 和 Haibin Zhang。2022。动态BFT基础。密码学电子预印本库（2022）。[43] 黑匣子101：了解事件数据记录器（EDR）。2020。Pushor Mitchell。https://www.pushormitchell.com/2020/11/black-box-101-understanding-event-data-recorders-edr/。[44] Muhammad El-Hindi、Carsten Binnig、Arvind Arasu、Donald Koss- mann 和 Ravi Ramamurthy。2019。BlockchainDB：基于区块链的共享数据库。VLDB基金会会刊12，11（2019），1597–1609。[45] Vasiliy Elagin、Anastasia Spirkina、Mikhail Buinevich 和 Andrei Vladyko。2020。车联网区块链应用的技术方面。《信息》11，10（2020），465。[46] Qi Feng、Debiao He、Sherali Zeadally、Muhammad Khurram Khan 和 Neeraj Kumar。2019。区块链系统隐私保护综述。《网络与计算机应用期刊》126（2019），45–58。[47] Michael J Fischer、Nancy A Lynch 和 Michael Merritt。1986。分布式共识问题的简易不可能性证明。《分布式计算》1，1（1986），26–39。[48] 福布斯。2020。GM Financial宣布与区块链初创公司 Spring Labs合作。https://www.forbes.com/sites / sarahhansen/2019/02/11/gm-financial-announces-collaboration-with-blockchain-startup-spring-labs/?sh=34403f8f31c9。[49] 福布斯。2020。梅赛德斯借助区块链推进Ambition2039计划。https://www.forbes.com/sites/darrynpollock/2020/01/31/mercedes-leveraging-blockchain-as-part-of-its-ambition2039-initiative/?sh=79f5c9464b8a。[50] 福布斯。2022。加利福尼亚州网联汽车数据隐私之争出现有趣转折。https://www.forbes.com/sites/stevetengler/20 22 05 17/ / / privacy-battle-over-connected-cars-takes-an-interesting-turn-in-california/?sh=23c35e164d72。[51] 福特。2020。福特研究表明，区块链、动态地理围栏和插电式混合动力厢式车有助于改善城市空气质量。https://media.ford.com/content/fordmedia/feu/en/news/2020/12/17/ford-study-shows-blockchain--dynamic-geofencing-and-plug-in-hybr.html。[52] Miguel Garcia、Alysson Bessani 和 Nuno Neves。2019。Lazarus：自动管理拜占庭容错系统中的多样性。载于第20届国际中间件会议论文集。241–254。[53] Mario H Castañeda Garcia、Alejandro Molina-Galan、Mate Boban、Javier Gozalvez、Baldomero Coll-Perales、Taylan Şahin 和 Apostolos Kousaridas。2021。5G NR车联网通信教程。IEEE《通信调查与教程》23，3（2021），1972–2026。[54] Yossi Gilad、Rotem Hemo、Silvio Micali、Georgios Vlachos 和 Nick- olai Zeldovich。2017。Algorand：为加密货币扩展拜占庭协议。载于第26届操作系统原理研讨会论文集。51–68。[55] 标普全球。2022年。保险公司翘首以待自动驾驶汽车之际，数据访问成为首要关切。https://www.spglobal.com/marketintelligence/en/news-insights/latest-news-headlines/data-access-among-top-concerns-as-insurers-await-autonomous-cars-69760098。[56] Rachid Guerraoui、Nikola Knežević、Vivien Quéma 和 Marko Vukolić。2010。接下来的700种BFT协议。载于第五届欧洲计算机系统会议论文集。363–376。[57] Rachid Guerraoui 和 André Schiper。2001。通用共识服务。IEEE《软件工程汇刊》27，1（2001），29–41。[58] Guy Golan Gueta、Ittai Abraham、Shelly Grossman、Dahlia Malkhi、Benny Pinkas、Michael Reiter、Dragos-Adrian Seredinschi、Orr Tamir，以及 Alin Tomescu。2019年。SBFT：一种可扩展的去中心化信任基础设施。载于2019年第49届IEEE/IFIP国际可靠系统与网络会议（DSN）。IEEE，568–580。[59] Lachlan J Gunn、刘健、Bruno Vavala和N Asokan。2019年。利用可信单调计数器增强推测式拜占庭容错的韧性。载于2019年第38届可靠分布式系统研讨会（SRDS）。IEEE，– 13309。[60] Suyash Gupta、Sajjad Rahnama、Jelle Hellings和Mohammad Sadoghi。 [无日期]。ResilientDB：全球规模的弹性区块链架构。《VLDB基金会会刊》13卷6期（[无日期]）。[61] Vikas Hassija、Vinay Chamola、韩光杰、Joel JPC Rodrigues和Mohsen Guizani。2020年。DAGIoV：一种利用有向无环图和博弈论实现车车通信的框架。IEEE车辆技术汇刊69卷4期（2020年），4182–4191。[62] HotStuff。2019年。Libhotstuff：一种通用的拜占庭容错状态机复制库，兼具模块化与简洁性。https://github.c om/hot-stuff/libhotstuff。[63] 侯若木、于海峰和Prateek Saxena。2022年。利用以吞吐量为中心的拜占庭广播，在区块链中容忍恶意多数派。载于2022年IEEE安全与隐私研讨会（SP）。IEEE，1263–1280。[64] Earl W Huff Jr、Siobahn Day Grady和Julian Brinnkley。2021年。告诉我需要知道什么：消费者对自动驾驶车辆信息透明度的诉求。载于《人因与工效学学会年会论文集》，第65卷。SAGE Publications Sage CA：Los Angeles, CA，327–331。[65] IBM。2020年。区块链驱动的自动驾驶汽车或可成为解决方案。https://www.ibm.com/blogs/blockchain/2020/04/blockchain-powered-autonomous-automobiles-can-be-the-answer/。[66] QAD Inc. 2020年。区块链改变汽车行业的五种方式。https://www.qad.com/blog/2020/03/5-ways-blockchain-is-changing-the-automotive-industry。[67] 什么是黑匣子？我的汽车有黑匣子吗？2022年。LALANDE PERSONAL INJURY LAWYERS。https://injured.ca/what-is-a-black-box-and-does-my-car-have-one/。[68] 蒋体刚、方华和王洪刚。2018年。基于区块链的车联网：分布式网络架构与性能分析。IEEE物联网期刊6卷3期（2018年），4640–4649。[69] Håvard Johansen、André Allavena和Robbert Van Renesse。2006年。Fireflies：为入侵容忍网络覆盖层提供可扩展支持。ACM SIGOPS《操作系统评论》40卷4期（2006年），3–13。[70] Rüdiger Kapitza、Johannes Behl、Christian Cachin、Tobias Dis- tler、Simon Kuhnle、Seyed Vahid Mohammadi、Wolfgang Schröder- Preikschat和Klaus Stengel。2012年。CheapBFT：资源高效的拜占庭容错。载于第7届ACM欧洲计算机系统会议论文集。295–308。[71] Ramakrishna Kotla、Lorenzo Alvisi、Mike Dahlin、Allen Clement和Edmund Wong。2007年。Zyzzyva：推测式拜占庭容错。载于第21届ACM SIGOPS操作系统原理研讨会论文集。45–58。[72] 丰田区块链实验室。2020年。加速区块链技术倡议与外部合作。https://global.toyota/en/newsroom/corporate/31827481.html。[73] 莱斯利·兰波特。2011年。简短公告：无领导者拜占庭Paxos。载于分布式计算国际研讨会。施普林格，141–142。[74] 莱斯利·兰波特和Michael Fischer。1982年。拜占庭将军与事务提交协议。技术报告。技术报告62，SRI国际。[75] 莱斯利·兰波特、Dahlia Malkhi和Lidong Zhou。2010年。重新配置状态机。ACM SIGACT新闻41卷1期（2010年），63–73。[76] 莱斯利·兰波特、Robert Shostak和Marshall Pease。2019年。拜占庭将军问题。载于《并发：莱斯利·兰波特的作品》。

203–226。[77] Dave Levin、John R Douceur、Jacob R Lorch和Thomas Moscibroda。2009年。TrInc：面向大型分布式系统的小型可信硬件。载于NSDI，第9卷。1–14。[78] Michael Lewis。2014年。《闪电男孩：华尔街的一场反抗》。WW Norton & Company。[79] Benoît Libert、Marc Joye和Moti Yung。2016年。分布式生成：具有短份额的完全分布式、非交互、自适应安全门限签名。《理论计算机科学》645（2016年），1–24。[80] LimeChain。 [无日期]。当今汽车行业的问题。https://lime
c
hain.tech/blockchain-use-cases/blockchain-auto
motive-industry/。[81] Shengyun Liu、Paolo Viotti、Christian Cachin、Vivien Quéma和Marko Vukolić。2016年。XFT：超越崩溃故障的实用容错。载于第12届USENIX操作系统设计与实现研讨会（OSDI 16）。485–500。[82] Jacob R Lorch、Atul Adya、William J Bolosky、Ronnie Chaiken、John R Douceur和Jon Howell。2006年。迁移复制有状态服务的SMART方式。载于2006年首届ACM SIGOPS/EuroSys欧洲计算机系统会议论文集。103–115。[83] J-P Martin和Lorenzo Alvisi。2004年。动态拜占庭存储框架。载于2004年国际可靠系统与网络会议。IEEE，325–334。[84] J-P Martin和Lorenzo Alvisi。2006年。快速拜占庭共识。IEEE《可靠与安全计算汇刊》3卷3期（2006年），202–215。[85] Andrew Miller、Yu Xia、Kyle Croman、Elaine Shi和Dawn Song。2016年。拜占庭容错协议中的蜜獾。载于2016年ACM SIGSAC计算机与通信安全会议论文集。31–42。[86] Achour Mostefaoui、Hamouma Moumen和Michel Raynal。2014年。在t< n/3条件下，采用O（n2）条消息实现无签名的异步拜占庭共识。载于2014年ACM分布式计算原理研讨会论文集。2–9。[87] MystenLabs。 [无日期]。Narwhal and Tusk。https://github.com/MystenL abs/narwhal。[88] 中本聪。2019年。比特币：一种点对点电子现金系统。技术报告。Manubot。[89] Ray Neiheiser、Miguel Matos和Luís Rodrigues。2021年。Kauri：通过流水线树形传播与聚合实现可扩展的拜占庭容错共识。载于ACM SIGOPS第28届操作系统原理研讨会论文集。35–48。[90] Joachim Neu、Ertem Nusret Tas和David Tse。2021年。潮起潮落协议：解决可用性与最终性困境。载于2021年IEEE安全与隐私研讨会（SP）。IEEE，446–465。[91] 彭博新闻。2022年。愤怒的特斯拉车主在上海车展上爬上车顶抗议。https://www.bloomberg.com/news/articles/20
21 04 19- - /
angry-tesla-owner-protests-atop-car-at-shanghai-auto- show#
xj4y7vzkg。[92] 美国州议会全国会议。2020年。自动驾驶车辆已颁布法规。https://www.ncsl.org/research/transportation/a
u
tono
mous-vehicles-self-driving-vehicles-enacted-legislation.aspx。[93] Jo-Ann Pattinson、Haibo Chen和Subhajit Basu。2020年。自动化车辆中的法律问题：审慎考量同意与交互式数字界面的潜在作用。《人文与社会科学通讯》7卷1期（2020年），1–10。[94] Li Peng、Wei Feng、Zheng Yan、Yafeng Li、Xiaokang Zhou和Shohei Shimizu。2021年。无需许可的区块链中的隐私保护：综述。《数字通信与网络》7卷3期（2021年），295–307。[95]《南华早报》。2022年。2021年上海车展特斯拉抗议事件：女子爬上车顶大喊后被安保人员拖走。https://www.scmp.com/news/people-culture/article/3130311/tesla-protest-shanghai-auto-show-2021-

[96]《华盛顿邮报》。2022年。自去年以来，特斯拉汽车启用自动驾驶辅助系统时发生的273起事故已有报告。https://www.washingtonpost.c
om/technology/2022/06/15/tesla-autopilot-crashes/.[97] Ji Qi、Xusheng Chen、Yunpeng Jiang、Jianyu Jiang、Tianxiang Shen、Shixiong Zhao、Sen Wang、Gong Zhang、Li Chen、Man Ho Au 等。2021年。Bidl：面向数据中心网络的高吞吐量、低延迟许可链框架。载于 ACM SIGOPS 第28届操作系统原理研讨会论文集。18–34。[98] 高通。2021年。通过蜂窝车联网实现车辆万物互联。。https://www.qualcomm.com/research/5g/cellular-v2x。[99]《通用数据保护条例》。2022年。GDPR：个人数据处理相关原则。https://gdpr-info.eu/art-5- gdpr/.[100] ResilientDB。 [不详。]。ResilientDB：可扩展的许可链架构。https://github.com/resilientdb/resilientdb.[101] Rodrigo Rodrigues、Barbara Liskov、Kathryn Chen、Moses Liskov 和 David Schultz。2010年。面向大规模可靠存储系统的自动重配置。IEEE《可靠与安全计算汇刊》9卷2期（2010年），145–158。[102] Mark Russinovich、Edward Ashton、Christine Avanessians、Miguel Castro、A
maury Chamayou、Sylvan Clebsch、Manuel Costa、Cédric Fournet、Matthew Kerner、Sid Krishna、Julien Maffre、Tho
mas Moscibroda、Kartik Nayak、Olya Ohrimenko、Felix Schuster、Roy Schwartz、Alex Shamis、Olga Vrousgou 和 Christoph M. Wintersteiger。2019年。CCF：用于构建机密可验证复制服务的框架。技术报告 MSR-TR-2019-16。微软。https://www.
microsoft.com/en-us/research/publication/ccf-a-fram
ework-for-building-confidential-verifiable-replicated-services/[103] Fred B Schneider。1984年。拜占庭将军实战：实现故障停止处理器。ACM《计算机系统汇刊》（TOCS）2卷2期（1984年），145–154。[104] Fred B Schneider。1990年。状态机方法：教程。容错分布式计算（1990年），18–41。[105] Alex Shamis、Peter Pietzuch、Burcu Canakci、Miguel Castro、Cédric Fournet、Edward Ashton、Amaury Chamayou、Sylvan Clebsch、Antoine Delignat-Lavaud、Matthew Kerner 等。2022年。IA-CCF：许可账本的个人可问责性。载于第19届 USENIX 网络系统设计与实现研讨会（NSDI 22）。467–491。[106] Victor Shoup。2000年。实用门限签名。载于密码技术理论与应用国际会议。施普林格，207–220。[107] Lorraine Sommerfeld。2021年。自动驾驶汽车即将上路，保险公司却尚未做好准备。https://driving.ca/column/lorrai
ne/autonomous-cars-are-on-their-way-and-insurance-companies-arent-ready。[108] Yee Jiun Song 和 Robbert van Renesse。2008年。Bosco：一步式拜占庭异步共识。载于国际分布式计算研讨会。施普林格，438–450。[109] Joao Sousa、Alysson Bessani 和 Marko Vukolic。2018年。面向 HyperLedger Fabric 区块链平台的拜占庭容错排序服务。载于2018年第48届 IEEE/IFIP可靠系统与网络国际会议（DSN）。IEEE，51–58。[110] Ion Stoica、Robert Morris、David Karger、M Frans Kaashoek 和 Hari Balakrishnan。2001年。Chord：面向互联网应用的可扩展对等查找服务。ACM SIGCOMM计算机通信评论31卷4期（2001年），149–160。[111] Florian Suri-Payer、Matthew Burke、Zheng Wang、Yunhao Zhang、Lorenzo Alvisi 和 Natacha Crooks。2021年。Basil：以 ACID（交易）打破拜占庭容错。载于 ACM SIGOPS 第28届操作系统原理研讨会论文集。1–17。[112] 特斯拉。 [不详。]自动驾驶辅助系统。https://www.tesla.com/autopilot.

ends- 女人 -d 粗布 d.

[113] Robin Vassantlal、Eduardo Alchieri、Bernardo Ferreira 和 Alysson Bessani。2022 年。COBRA：面向机密拜占庭容错服务的动态主动秘密共享。载于 *2022*年 *IEEE*安全与隐私研讨会（*SP*）。IEEE计算机学会，1528–1528。[114] Peter Virk。2021年。网联汽车需要数据透明度。https: // www.autonews.com/commentary/connected-cars-require-data-transparency。[115] 大众汽车。2019年。利用区块链保护人与环境。https://www.volkswagenag.com/en/news/stories/2019/04/protecting-people-and-the-environment-with-blockchain.htm l#。[116] 大众汽车。2021年。大众汽车试点利用区块链将电动汽车接入电网。https://cointelegraph.com/news/volkswagen-pilots-blockchain-to-integrate-electric-vehicles-with-power- grids。[117] Zhuolun Xiang、Dahlia Malkhi、Kartik Nayak 和 Ling Ren。2021年。增强拜占庭容错复制的容错能力。载于*2021*年 *IEEE*第*41*届国际分布式计算系统会议（*ICDCS*）。IEEE，205–215。[118] Lei Yang、Seo Jin Park、Mohammad Alizadeh、Sreeram Kannan 和 David Tse。2021。DispersedLedger：可变带宽网络上的高吞吐量拜占庭共识。*arXiv* 预印本 *arXiv:2110.04371*（2021）。[119] Maofan Yin、Dahlia Malkhi、Michael K. Reiter、Guy Golan-Gueta 和 Ittai Abraham。2019。HotStuff：具备线性性和响应性的拜占庭容错共识。载于*2019*年*ACM*分布式计算原理研讨会论文集，*PODC 2019*，安大略省多伦多，*2019*年*7*月*29*日至*8*月*2*日。ACM，347–356。[120] Yong Yuan 和 Fei-Yue Wang。2016。迈向基于区块链的智能交通系统。载于*2016*年*IEEE*第*19*届智能交通系统国际会议（*ITSC*）。IEEE，2663–2668。[121] Gengrui Zhang 和 Hans-Arno Jacobsen。2021。Prosecutor：一种高效的拜占庭容错共识算法，通过行为感知惩罚机制应对拜占庭攻击。载于第*22*届国际中间件会议论文集。52–63。[122] Yunhao Zhang、Srinath T. V. Setty、Qi Chen、Lidong Zhou 和 Lorenzo Alvisi。2020。无需拜占庭寡头统治的拜占庭有序共识。载于第*14*届*USENIX*操作系统设计与实现研讨会，*OSDI 2020*，线上会议，*2020*年*11*月*4*日至*6*日。USENIX协会，633–649。