Operate 字段包括 new 和 back 两个方式, new 表示直接在 VC 服务器当前状态的数据库进行 RPKI 缓存更新, back 表示需要 VC 服务器回退到之前版本的数据库.

As 字段用以实现面向不同的 AS 的本地化信息控制视图, 管理者可通过修改 as 字段, 构建针对不同 AS 的本地化信息控制视图.

3.2 VC 服务器

在本文的设计方案中, 共有 3 个实体, RP 服务器, VC 服务器和路由器, 由于 RP 服务器与路由器的设计在工业上已有实现, 故本文设计方案仅讨论分发架构中 VC 服务器的设计.

3.2.1 VC 服务器功能模块

为了能够安全高效的进行 RPKI 缓存分发, 结合第 2 节中的设计思路, VC 服务器应具有五种功能模块. (1) 数据包构建模块: 该模块用于将 RPKI 缓存描述为如 2.3 所述格式, 并通过添加头部控制信息, 构建如 3.1 所述的数据包; (2) 传输模块: 根据头部控制信息中的 as 字段将 RPKI 缓存通过 HTTPS 协议分发到对应 as 区域中的 VC 服务器, 并通过多线程实现一次性多目标分发; (3) 数据解析模块: 该模块用于解析收到的 RPKI 缓存数据包, 通过头部控制信息对 RPKI 缓存进行相应的处理; (4) 重传模块: 当 VC 服务器收到不完整数据包或者发现有数据包遗漏时, 该模块会向其上级 VC 服务器请求重新获取 RPKI 缓存更新, 并提供相应的控制信息. 当 VC 服务器接收到下级 VC 服务器的重发请求时, 将根据其提供的控制信息, 调用数据包构建模块构建数据包, 并通过传输模块进行信息的传输. 如图 6 所示.

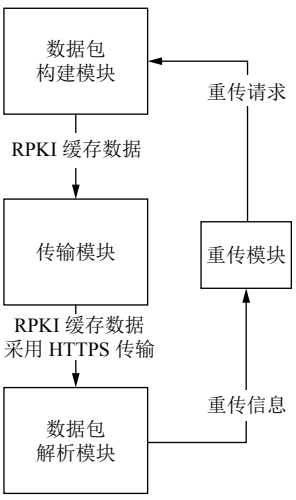

图 6 VC 功能模块示意图

3.2.2 安全保护与可靠性

由于 HTTPS 的安全加密通道只能够保证传输过程中的数据安全. 在本文的设计方案中还建立了 IP 访问控制和数据完整性检查两种机制进行安全性保护, 并通过重传机制增强分发架构的可靠性.

1) IP 访问控制

VC 服务器在接受数据包之前, 首先会对传输数据的 IP 进行检查, 如果该 IP 不合法, VC 服务器会直接抛弃该数据包, 不对其中的数据进行解析. 该机制可以保证 VC 服务器接受的 RPKI 缓存数据的来源是可靠的.

2) 数据完整性检查

VC 服务器在进行数据包解析之前, 首先会根据 sha1 字段中的数据对数据包进行完整性校验, 如果校验结果不相符, 则抛弃该数据包, 并通过重传请求模块向上级 VC 服务器申请数据包的重传.

3) 重传机制

当数据包未能通过数据完整性检测或者数据解析过程中发现 newversion 字段与数据库中的最新版本相差大于 1, VC 服务器就会启用重传机制, 向上级 VC 服务器发出请求重新获取数据包.

4 实验与分析

4.1 RPKI 缓存分发实验

RPKI 缓存分发流程如图 7 所示, VC 服务器会根据需要分发的 RPKI 缓存更新, 利用 3.1 所述数据结构构建分发数据包, 通过 HTTPS 协议将数据包分发到指定 AS 处的 VC 服务器. VC 服务器会对数据包进行检测, 如果验证失败则触发重传请求, 请求重传 RPKI 数据包. 之后, 根据数据包进行相应的处理和更新.

4.1.1 RPKI 缓存分发实验设计与结果分析

实验中使用五台物理机模拟传输架构进行实验, 将五台物理机分别标号为 1–5, 其中 1 号机 7999 端口作为 RP 服务器, 8000 端口作为 VC 服务器, 2–5 号机依次作为不同层级的 VC 服务器.

实验使用 JSON 格式的数据包进行传输测试. 通过输出时间戳 (毫秒精度) 计算发起连接到通信完成之间的时间差, 从而得到实际过程中的时间效率. 本实验通过 3 个端口 (8000–8002) 得到每个层级分发用时的平均值 (同时测试多线程分发) 并计算总分发用时, 如图 8 所示.

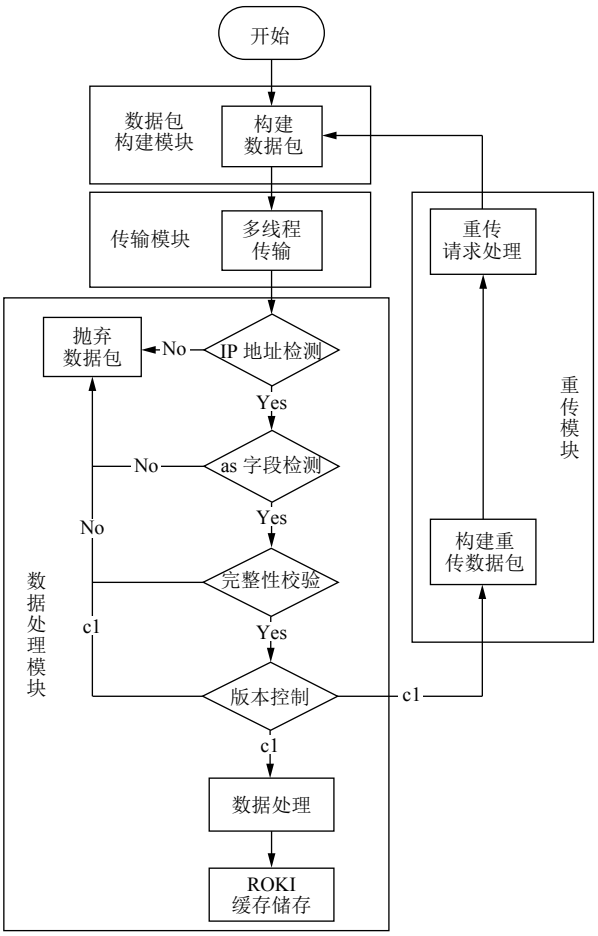

图 7 RPKI 缓存分发流程图

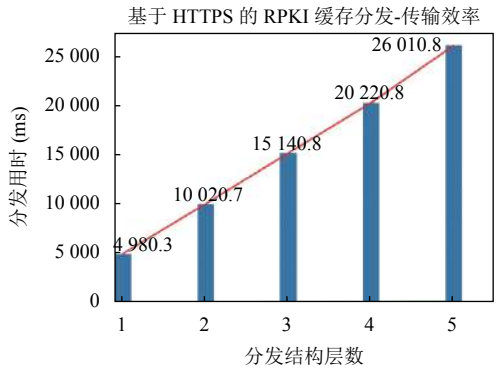

图 8 RPKI 缓存分发用时图

图中的 X 轴表示分发结构中 VC 服务器的传输层数, Y 轴表示总分发用时平均值 (单位: ms), 实际测量得到, 5 层分发结构仅用时 26 010.8 ms(26.0108 s). 在实际部署场景中, 该架构能够迅速稳定的分发 RPKI 缓存, 可以满足实际运行的需要.

本文还对核心模块进行单独测试, 以下给出运行结果.

4.1.2 数据包构建模块

数据包构建模块通过用户输入或者根据重传请求构建数据包, 以 JSON 格式为例.

运行结果如图 9 所示.

图 9 数据包构建模块测试结果

4.1.3 传输模块

传输模块根据数据包头部 AS 字段获取目标 AS 的 IP 地址, 并通过多线程的方式分发 RPKI 缓存, 服务器端运行结果如图 10 所示.

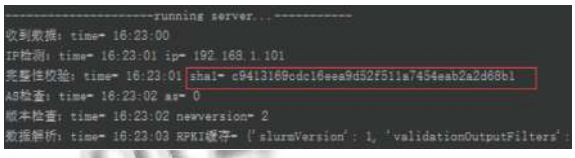

图 10 服务器端运行结果

4.1.4 数据处理模块

数据处理模块功能主要分为以下两部分:

1) IP 访问控制, 完整性检测与版本控制

2) 数据处理与储存

运行结果如图 11 所示.

newversion	operate	asn	prefix	prefixlength
2	Filters		1.10.10.0	24
2	Filters	999996	1.9.6.0	24
2	Filters	1	10.89.56.0	24
2	Filters	483	144.96.69.(	24
2	Assertion	199998	19.99.98.0	24

图 11 数据处理模块运行结果

4.2 性能测试

为了准确评估本方案的实际运行效果, 还设计了

对比试验. 第一组为使用 RTR 协议进行 RPKI 缓存的分发. 第二组为使用 HTTPS 协议进行 RPKI 缓存的分发. 由于架构和数据处理操作相同, 本实验只对 RTR 协议和 HTTPS 协议的传输时间进行测试 (不计算数据处理用时), 本实验根据 RTR 协议逐条传输的特点, 测试在五种不同数目的 RPKI 更新缓存数据 (分别为 10 条, 20 条, 50 条, 100 条, 200 条) 的情况下, RTR 协议与 HTTPS 协议的传输时间对比. 实验结果如图 12 所示.

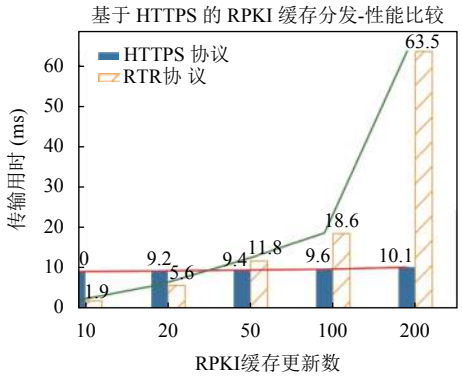

图 12 性能比较结果

从图 11 可以看出, 当 RPKI 缓存更新数较少的时候, RTR 协议传输效率比 HTTPS 协议高, 但是随着缓存更新数的增加, RTR 协议传输时间也显著增加, 而 HTTPS 协议并无明显变化. 在 RPKI 缓存更新数达到 200 时, RTR 协议用时为 HTTPS 协议的 6.28 倍. 因此在实际部署环境中, 使用 HTTPS 协议可以使得传输效率更加稳定.

此外, 文献[9]提到在 RTR 协议中客户端每间隔一小时通过服务器端获取一次数据. 在多层分发架构中, 客户端获取数据的周期叠加会使得分发时间长达数小时. 而本文提出的基于 HTTPS 协议的分发架构能够使得分发时间不会受到周期叠加的影响, 可以高效分发 RPKI 缓存.

5 结束语

随着互联网时代的进一步发展, RPKI 的部署势在必行, 也获得了国内外许多组织的大力推行. 但是在实际部署过程中还有许多问题需要进行研究. 在这种背景下, 本文利用 JSON 化的 RPKI 验证缓存数据实现了一种基于 HTTPS 的缓存更新机制. 本文首先说明了现有 RPKI 架构的难以应用于实际部署的缺陷, 并给出了本方案的设计考量, 然后给出了设计的完整内容和具体细节. 在此基础上本文用 python 对该设计进行了实现并与 RTR 协议进行对比.

RPKI 作为路由安全领域的热点话题, 既是为互联网运行保驾护航的重要举措, 也是互联网进一步发展的基石. 因此如何将 RPKI 更好的部署在实际场景中也是一个重要的探索方向和研究内容.

参考文献

1 Kent S, Lynn C, Seo K. Secure border gateway protocol (S-BGP). IEEE Journal on Selected Areas in Communications, 2000, 18(4): 582–592. [doi: 10.1109/49.839934]

2 Lepinski M, Kent S. An infrastructure to support secure internet routing. RFC 6480. 2012. [doi: 10.17487/RFC6480]

3 Reynolds MC, Kent S. A high performance software architecture for a secure internet routing PKI. Proceedings of 2009 Cybersecurity Applications & Technology Conference for Homeland Security. Washington, DC, WA, USA. 2009. 49–53.

4 许圣明, 马迪, 毛伟, 等. 基于有序哈希树的 RPKI 资料库数据同步方法. 计算机系统应用, 2016, 25(6): 141–146.

5 Weiler S, Ward D, Housley R. The rsync URI scheme. RFC 5781. 2010.

6 司昊林, 马迪, 毛伟, 等. RPKI 增量同步 Delta 协议的形式化检测与实现. 计算机系统应用, 2018, 27(11): 1–8. [doi: 10.15888/j.cnki.csa.006562]

7 Mandelberg D, Bruijnzeels T, Muravskiy O, *et al*. The RPKI repository delta protocol. RFC 8182. 2017. [doi: 10.17487/RFC8182]

8 安春林, 马迪, 王伟, 等. 基于哈希表的 RPKI 证书验证优化方法. 计算机系统应用, 2018, 27(2): 132–137. [doi: 10.3969/j.issn.1003-3254.2018.02.022]

9 Bush R, Austein R. The resource public key infrastructure (RPKI) to router protocol, RFC 6810. 2013. [doi: 10.17487/RFC6810]

10 胡军. 复杂网络下多服务注册中心部署策略研究[硕士学位论文]. 重庆: 重庆大学, 2011.

11 Ma D, Mandelberg D, Bruijnzeels T. Simplified local internet number resource management with the RPKI (SLURM), RFC 8416. 2018. [doi: 10.17487/RFC8416]

12 Hoffman P, McManus P. DNS queries over HTTPS (DoH), RFC 8484. 2018. [doi: 10.17487/RFC8484]

13 Hoffman P. Representing DNS messages in JSON, RFC 8427. 2018. [doi: 10.17487/RFC8427]