Package Relay & TRUC

Johnny Santos

The "problem"

Bitcoin has a "scaling" problem. One that's there by **design**.

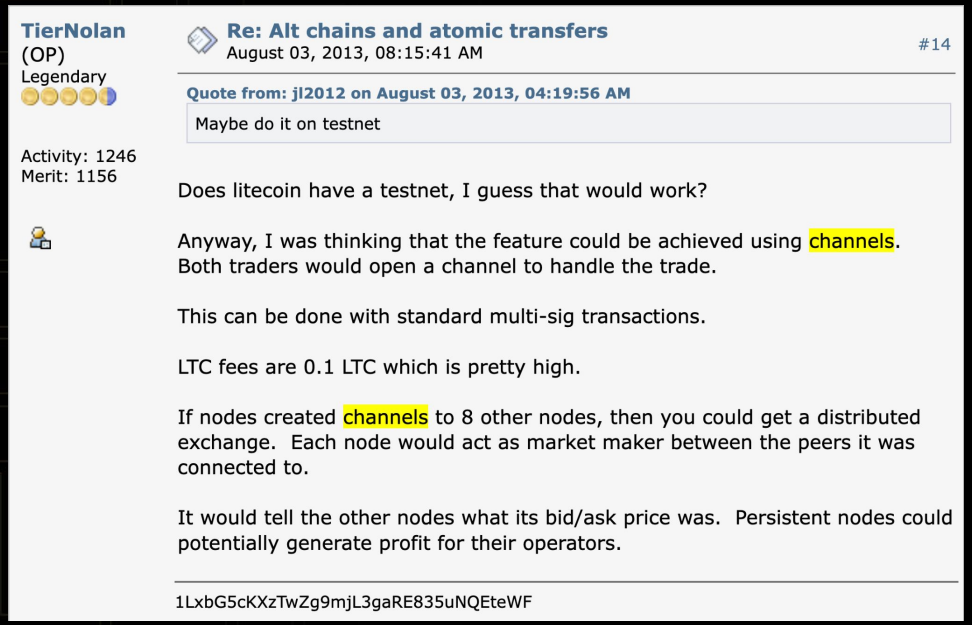
Multiple ideas get float around since the Nakamoto era.

One in particular receives multiple iterations and variations: **payment channels**. Another layer that build upon the security model of the base layer.

A bit of history

- 2010: Became clear that scaling Bitcoin would have to come from another layer
- 2012: after 1st halving adoption surged. Reigniting scaling talks.

A bit of history



TierNolan
(OP)
Legendary

Activity: 1246
Merit: 1156

Re: Alt chains and atomic transfers
August 03, 2013, 08:15:41 AM #14

Quote from: jl2012 on August 03, 2013, 04:19:56 AM

> Maybe do it on testnet

Does litecoin have a testnet, I guess that would work?

Anyway, I was thinking that the feature could be achieved using channels. Both traders would open a channel to handle the trade.

This can be done with standard multi-sig transactions.

LTC fees are 0.1 LTC which is pretty high.

If nodes created channels to 8 other nodes, then you could get a distributed exchange. Each node would act as market maker between the peers it was connected to.

It would tell the other nodes what its bid/ask price was. Persistent nodes could potentially generate profit for their operators.

1LxbG5cKXzTwZg9mjL3gaRE835uNQEteWF

2013: TierNolan suggests contracts to allow cross chain swaps and channels

A bit of history

- 2010: Became clear that scaling Bitcoin would have to come from another layer
- 2012: after 1st halving adoption surged. Reigniting scaling talks.
- 2013: TierNolan suggests contracts to allow cross chain swaps and channels
- 2014: proposals were divided between growing the base layer and building on second layers
- 2016: LN paper gets released with the internals of the protocol. Multiple implementations start to develop soon after.
- 2018: LN launches on mainnet (lnd, c-ln)

The problem reshapes

LN starts to get used.

Transaction pinning attack creates havoc in channel management leading to issues and potential loss of funds.

LN channels rely on time-sensitive penalty/justice transactions to enforce fairness. If one party broadcasts a cheating pre-signed state, the counterparty must respond quickly with a penalty transaction to claim funds.

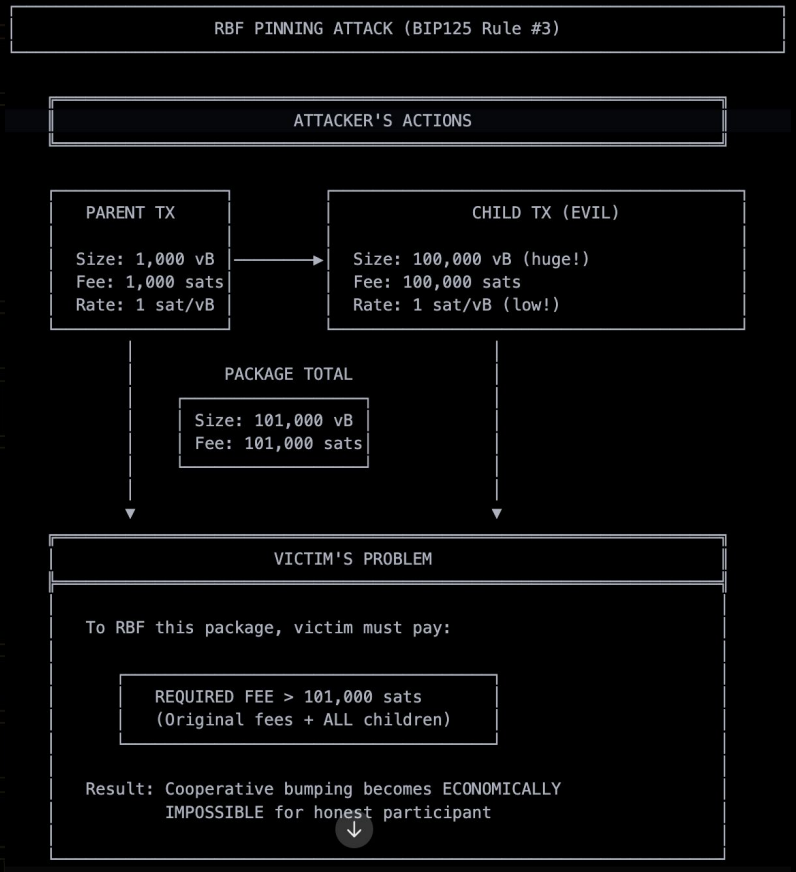
What exactly is pinning

Abusing transaction replacement rules to make the other party (in a multi party contract - LN for instance) pay excessive or prohibitively fees. This is especially dangerous when we use pre-signed transactions (justice tx).

Pinning can take many forms:

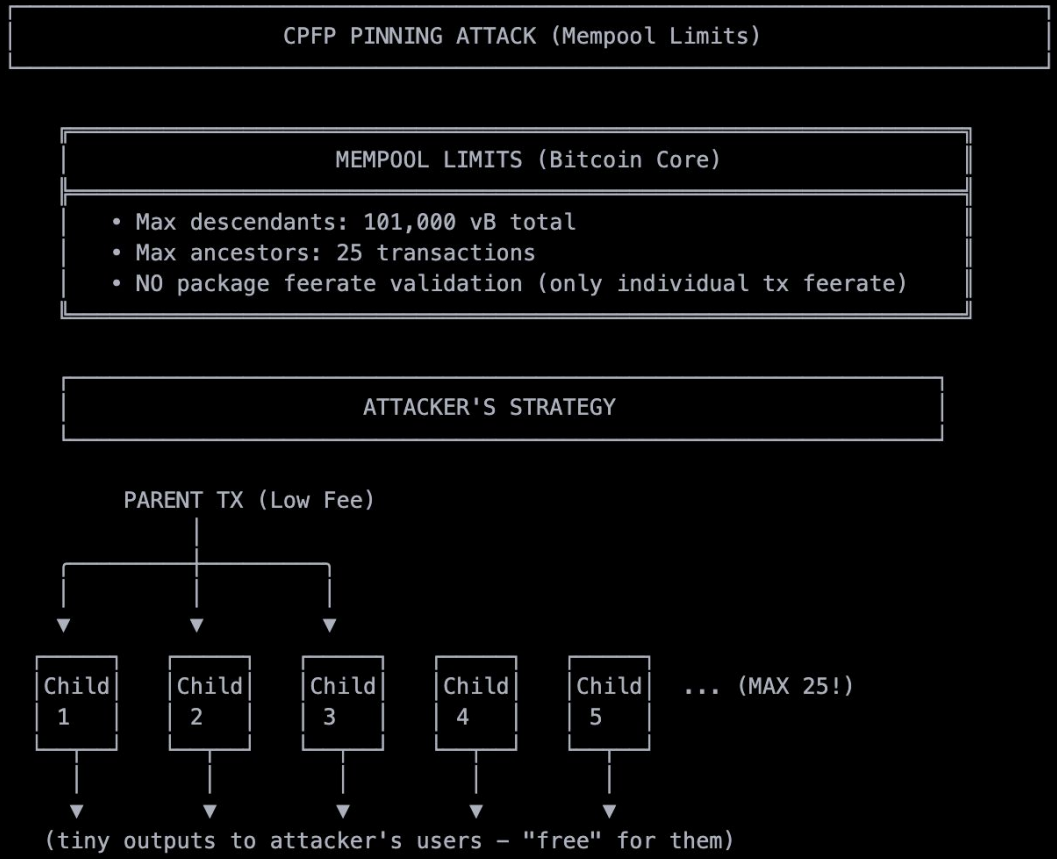
- **CPFP**: adding the max allow children before the other party
- **RBF**: abusing rule #3 that requires the replacement to pay MORE than the absolute total of the previous tx and its descendents

What exactly is pinning



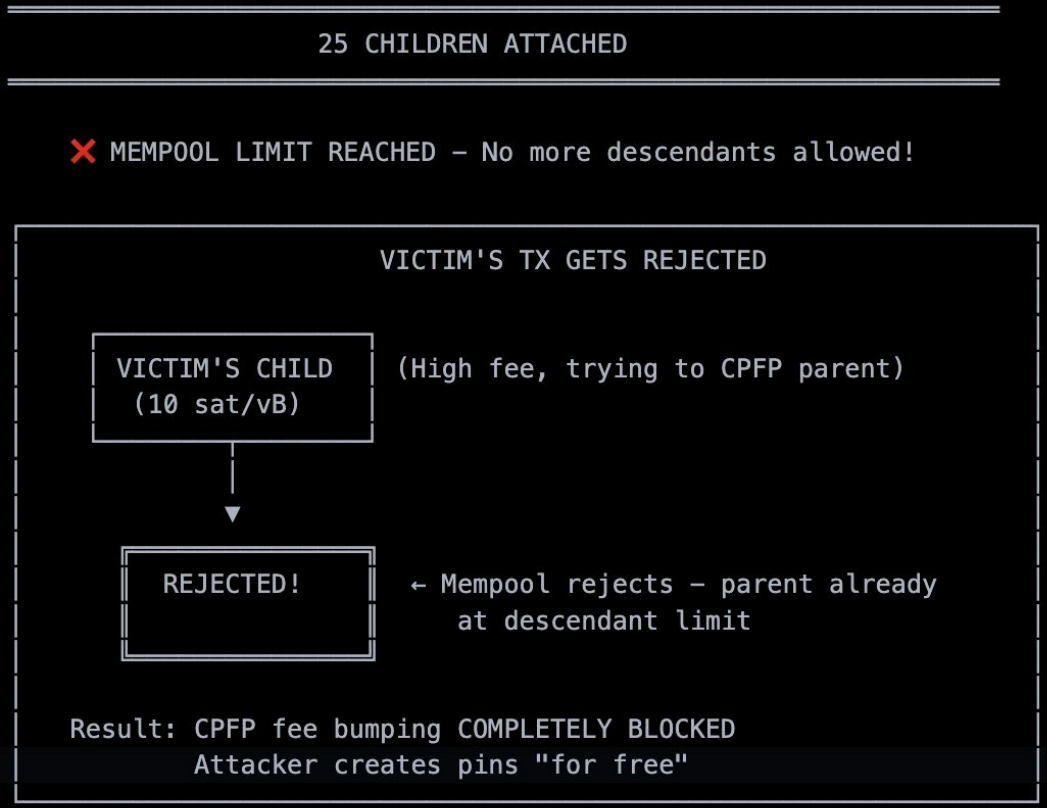
vinteum

What exactly is pinning



vinteum

What exactly is pinning



vinteum

1st attempts (2018-19)

Anchor Outputs: tiny output on the pre-signed transaction that allows the user to CPFP directly the target transaction to prioritize it. Prev. known as *simplified commitments*. Provides a hook for:

CPFP Carve-Out: extra rule to allow the 26th transaction (in the cluster) as a special exception to size rules if it attaches directly to the target transaction (only 1 ancestor allowed) and pays minimum fees.

1st attempts (2018-19)

Anchor Outputs: works sub optimally for CPFP Carve-Out. *Still needs package relay to be optimal.*

CPFP Carve-Out: works okay but users still have to pay a big fee. Not as big to be economically infeasible but big.

Still failed to prevent prohibitively costs in times where the mempool was overloaded.

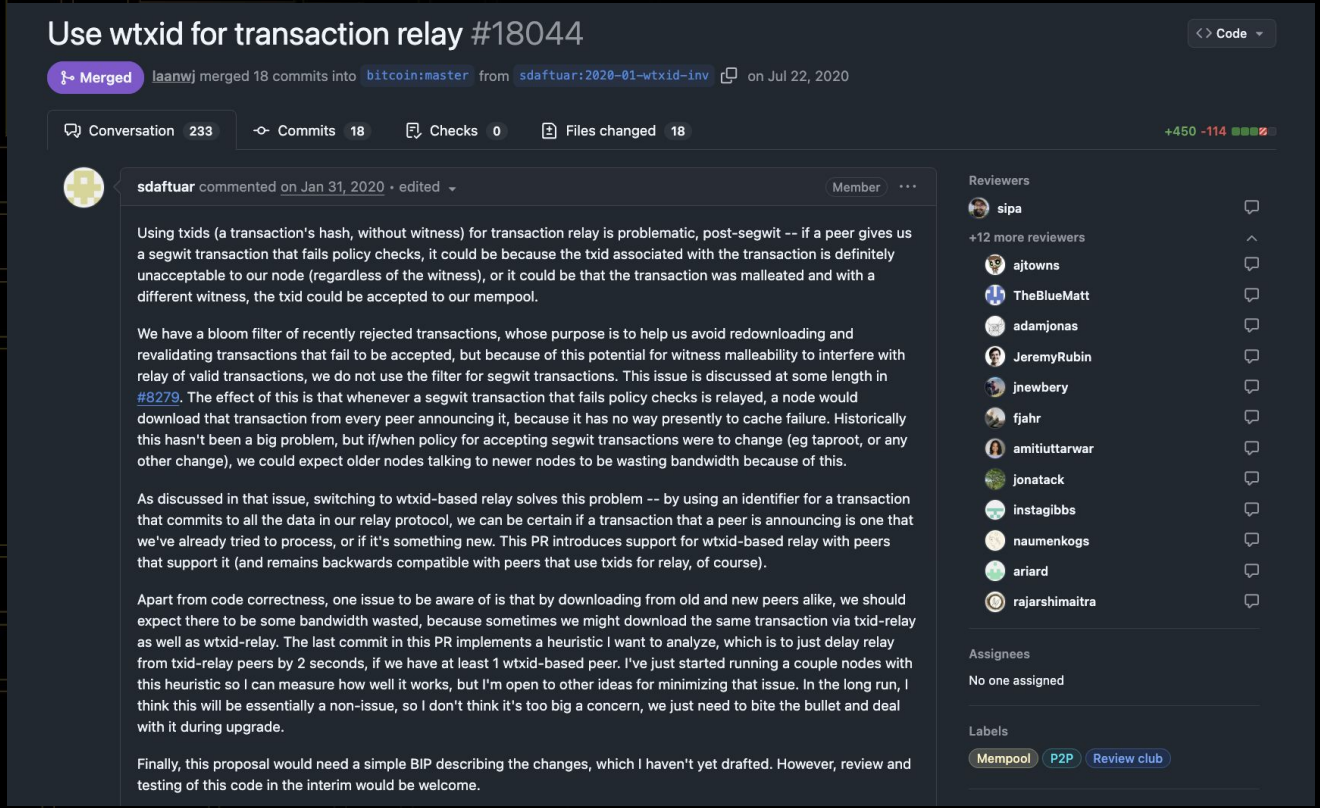
Yet another problem (2020)

Relays depend on the *malleable* **txid** and that creates havoc on the relay of time sensitive LN transactions.

That gets solved with **BIP339** using wtxid as the default in p2p messages.

Still no resolution to topological transaction evaluation as candidate for block mining.

Yet another problem (2020)



Use wtxid for transaction relay #18044

Code

Merged laanwj merged 18 commits into bitcoin:master from sdaftuar:2020-01-wtxid-inv on Jul 22, 2020

Conversation 233 Commits 18 Checks 0 Files changed 18 +450 -114

sdaftuar commented on Jan 31, 2020 • edited Member

Using txids (a transaction's hash, without witness) for transaction relay is problematic, post-segwit -- if a peer gives us a segwit transaction that fails policy checks, it could be because the txid associated with the transaction is definitely unacceptable to our node (regardless of the witness), or it could be that the transaction was malleated and with a different witness, the txid could be accepted to our mempool.

We have a bloom filter of recently rejected transactions, whose purpose is to help us avoid redownloading and revalidating transactions that fail to be accepted, but because of this potential for witness malleability to interfere with relay of valid transactions, we do not use the filter for segwit transactions. This issue is discussed at some length in #8279. The effect of this is that whenever a segwit transaction that fails policy checks is relayed, a node would download that transaction from every peer announcing it, because it has no way presently to cache failure. Historically this hasn't been a big problem, but if/when policy for accepting segwit transactions were to change (eg taproot, or any other change), we could expect older nodes talking to newer nodes to be wasting bandwidth because of this.

As discussed in that issue, switching to wtxid-based relay solves this problem -- by using an identifier for a transaction that commits to all the data in our relay protocol, we can be certain if a transaction that a peer is announcing is one that we've already tried to process, or if it's something new. This PR introduces support for wtxid-based relay with peers that support it (and remains backwards compatible with peers that use txids for relay, of course).

Apart from code correctness, one issue to be aware of is that by downloading from old and new peers alike, we should expect there to be some bandwidth wasted, because sometimes we might download the same transaction via txid-relay as well as wtxid-relay. The last commit in this PR implements a heuristic I want to analyze, which is to just delay relay from txid-relay peers by 2 seconds, if we have at least 1 wtxid-based peer. I've just started running a couple nodes with this heuristic so I can measure how well it works, but I'm open to other ideas for minimizing that issue. In the long run, I think this will be essentially a non-issue, so I don't think it's too big a concern, we just need to bite the bullet and deal with it during upgrade.

Finally, this proposal would need a simple BIP describing the changes, which I haven't yet drafted. However, review and testing of this code in the interim would be welcome.

Reviewers
sipa
+12 more reviewers
ajtowns
TheBlueMatt
adamjonas
JeremyRubin
jnewbery
fjahr
amitiuttarwar
jonatack
instagibbs
naumenkogs
ariard
rajarshimaitra

Assignees
No one assigned

Labels
Mempool P2P Review club

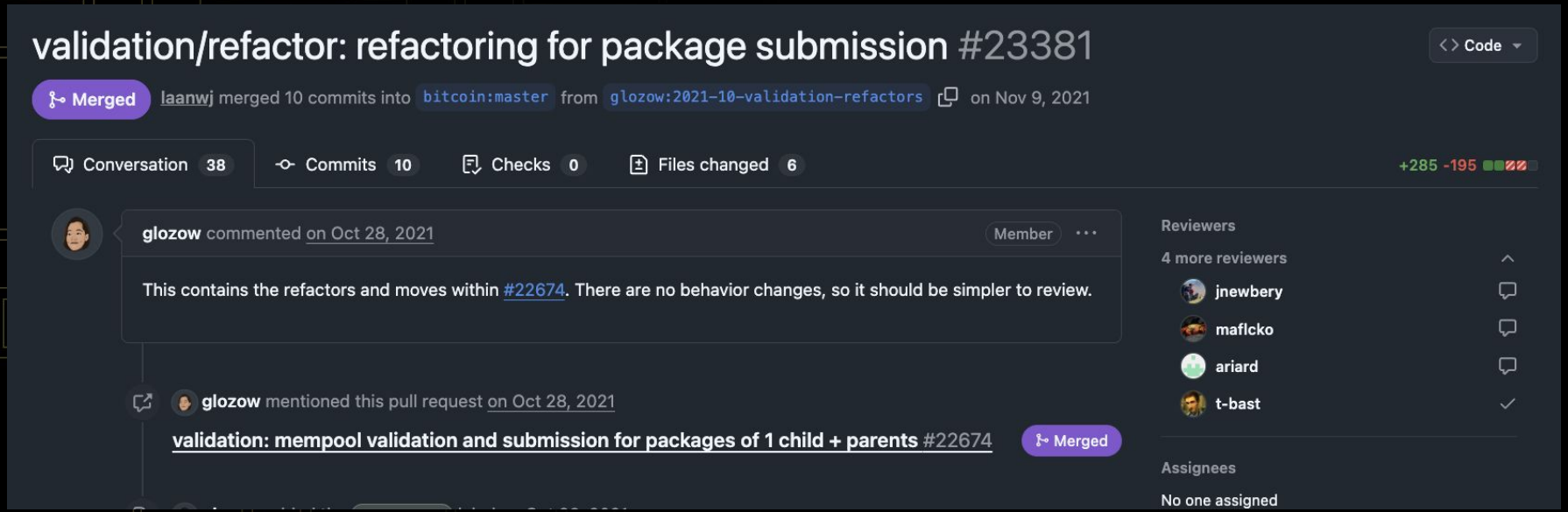
vinteum

Package relay prep (2021)

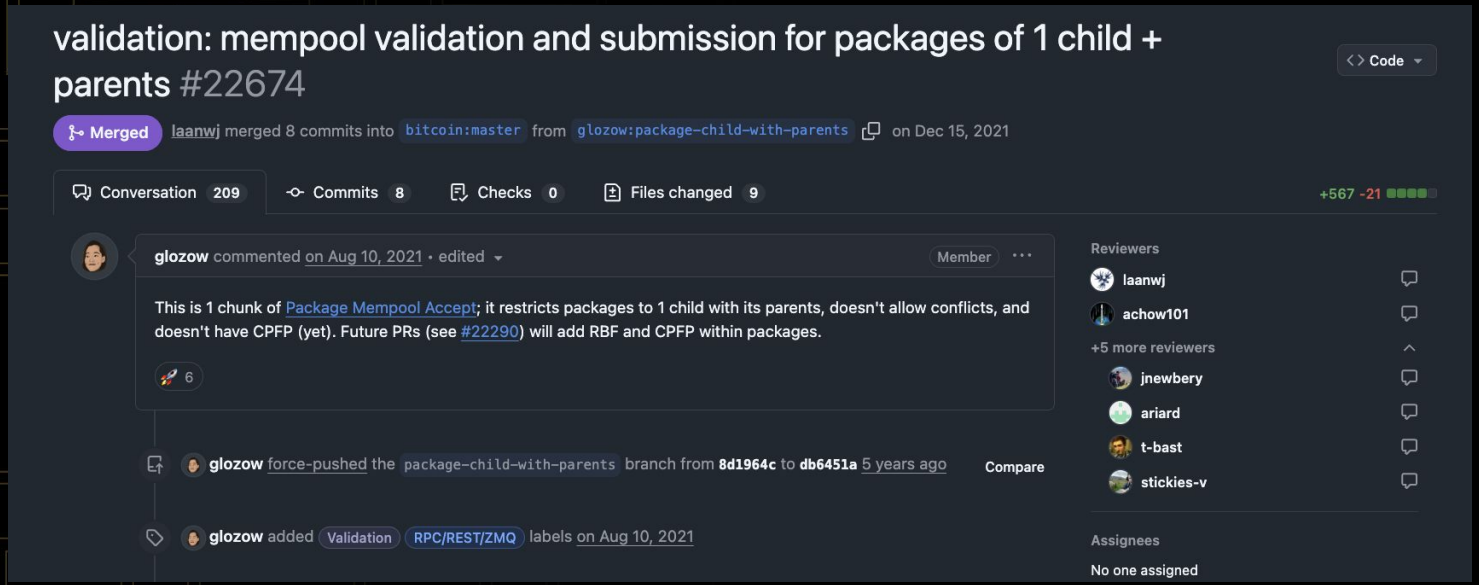
Pull requests start to get reviewed and merged on bitcoin core to allow package relay.

This takes a long time as it affects multiple submodules of the system.

Package relay prep (2021)



Package relay prep (2021)



vinteum

Package relay prep (2022)

BIP 331: Ancestor Package Relay

2022-08-08 View on GitHub

```
BIP: 331
Layer: Peer Services
Title: Ancestor Package Relay
Authors: Gloria Zhao <gloriajzhao@gmail.com>
Status: Draft
Type: Specification
Assigned: 2022-08-08
License: BSD-3-Clause
Discussion: 2022-05-17 https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2022-May/020493.html [bitco
```

Abstract

Peer-to-peer protocol messages enabling nodes to request and relay the unconfirmed ancestor package of a given transaction, and to request and relay transactions in batches.

Motivation

Propagate High Feerate Transactions

Since v0.13, Bitcoin Core has used ancestor packages instead of individual transactions to evaluate the incentive compatibility of transactions in the mempool [1] and selecting them for inclusion in blocks [2]. Incentive-compatible mempool and miner policies help create a fair, fee-based market for block space. While miners maximize transaction

vinteum

Package relay prep (2022)

Package Relay Project Tracking #27463

New issue

Closed

glozow opened on Apr 14, 2023 · edited by glozow Edits Member

Tasks and PRs

- [x] **(1) multi-parent-1-child package validation**

What we get: the ability to validate multiple transactions, including CPFP of transactions below the mempool minimum feerate. An RPC to submit things locally.

- Enable validation of multiple transactions in MemPoolAccept
  - [x] Dependency: refactor: return MempoolAcceptResult from ATMP #21062
  - [x] Dependency: validation/refactor: refactoring for package submission #23381
  - [x] Main feature 1/2: rpc/validation: enable packages through testmempoolaccept #20833
  - [x] Main feature 2/2: mempool/validation: mempool ancestor/descendant limits for packages #21800
  - [x] Followup: package testmempoolaccept followups #22084
- Enable package CPFP
  - [x] Main feature 1/2: validation: mempool validation and submission for packages of 1 child + parents #22674
  - [x] Followup: docs / fixups from RBF and packages #24310
  - [x] Followup: validation: followups for de-duplication of packages #23804

Assignees

No one assigned

Labels

Brainstorming Feature P2P TX fees and policy Tracking Issue

Type

No type

Projects

Package Relay + V3 Project Tracking

Status Done: Mempool

Milestone

No milestone

Relationships

vinteum

Package relay prep (2022)

- Enable package CPFP
  - [x] Main feature 1/2: validation: mempool validation and submission for packages of 1 child + parents #22674
  - [x] Followup: docs / fixups from RBF and packages #24310
  - [x] Followup: validation: followups for de-duplication of packages #23804
  - [x] Main feature 2/2: policy / validation: CPFP fee bumping within packages #24152
  - [x] short term bug fix: avoid the risk of below-minrelaytxfee transactions hanging around forever in the mempool mempool: disallow txns under min relay fee, even in packages #26933
- RPC access
  - [x] add RPC (-regtest only) for testing package policy #24836
  - [x] validation, bugfix: provide more info in *MempoolAcceptResult #26646
  - [x] rpc: allow submitpackage to be called outside of regtest #27609
  - [x] bugfix, Change up submitpackage results to return results for all transactions #28848
  - [x] RPC: Add maxfeerate and maxburnamount args to submitpackage #28950
  - [x] Followup: 28950 followups #29722
  - [x] Followup: AcceptMultipleTransactions: Fix workspace not being set as client_maxfeerate failure #29735

Package relay prep (2022)

- Fuzzing and bug fixes
  - [x] Add package evaluation fuzzer #28450
  - [x] Fuzz: Check individual and package transaction invariants #28764
  - [x] fuzz: Minor improvements to tx_package_eval target #28825
  - [x] Bug fix: validation: fix coins disappearing mid-package evaluation #28251
  - [x] Bug fix: Fix virtual size limit enforcement in transaction package context #28471
  - [x] Bug fix: Remove MemPoolAccept::m_limits to avoid mutating it in package evaluation #28472

- [x] **(2) Policy-Related Features: TRUC and limited package RBF**

What we get: an opt-in policy for anti-pinning in single transaction or 1-parent-1-child package scenarios. Also, package CPFP of 0-fee parent and package RBF for restricted topologies prior to cluster mempool.

- [x] **(2a) Topologically Restricted Until Confirmation (v3) transaction policy**

Package Relay (2022)

A feature proposal that is part of the **BIP331** where it defines p2p messages and recommends how to handle unconfirmed transactions.

The proposal allows for the submission to of topologically cohesive packages of unconfirmed transactions to be processed as one instead of being evaluated individually.

As of now the BIP still not fully implemented as it cover many topics, including even changes outside of the protocol (mempool, unconfirmed tx handling, and so on).

TRUC / v3 (2024)

TRUC (*Topologically Restricted Until Confirmation*) or transaction **version 3** are *superset* of rules for handling and relaying transactions that are topologically aware, meaning, the validation depends not on just one transaction alone.

This makes fee bumping more predictable and reliable, orphan transaction finding easier, and **prevent undesired behavior in multi party contracts**.

It is defined on **BIP431** and builds upon the groundwork set by package relay and its BIP.

Today

Package relay is partially implemented with a focus on **1p1c** (cluster size 2). It solves the problem of atomic relay of clusters of transactions. There are missing p2p messages.

Malleable txids don't affect relays anymore.

TRUC is not fully adopted (opt-in) but accepted as standard. Still needs implementation of p2p negotiation, messages, etc.

Depends on the adoption of v3 and cluster mempools. To expand and complete implementation.

Takeaway

Bitcoin is much more capable, resilient and prepared for numerous attacks focusing on multi party contracts to allow better scalability.

At the same time there's still a lot of work to do.

Related work

We can't cover all the related work to understand the big picture of the anti-pinning work allowing Bitcoin L2s to work optimally, so here's all (non exhaustive) for further research:

- Pay to Anchor (P2A) / Ephemeral Anchors
- Test Package Accept (RPC)
- Fee Sponsorship (consensus)
- Mempool policies: sibling eviction, cluster mempool,

Sources

- bitcoinops.org
- grokipedia.com
- bitcointalk.org

Learn more:

vinteum

Obrigado.

Johnny Santos @johnnyasantos

johnnysantos@protonmail.com