Prioritization of async payments

- There is a question of how highly to prioritize async payments because phones may give background apps some CPU time, actually. Waiting on some wallets to take measurements here.
- No matter what, async payments would be nice to have, as phones can die or be in airplane mode.

The problem we're trying to solve: LSPs maybe shouldn't be trusted to not issue the same invoice twice, because it would allow them to steal the payment. Hence, we want to discuss using offers such that the sender can request a static keysend bolt12 invoice from the receiver's LSP. (Whether PTLCs would solve reusable invoices is currently an open question, see below.)

- Rusty: if we *did* trust the LSP, at least you'd be trusting them to not steal your $$, not some random node on the network

PTLCs scheme to allow LSPs to issue invoices

- Is there a scheme where you can make some info come in to generate the preimage? so while you can prove it has been authorized, can't prove it's been authorized for this particular case.
- You can have this with keysend – you have assurance that the payment got to the final recipient
- Rusty: there have been some schemes with PTLCs with keysend-like behavior
- Matt: haven't spent time on it, would be interesting to try to document that.
- M: could prob do keysend + a nonce and the time you sent it and the amt, and can prove "i am one of the ppl who sent this much at this time" or sth like that. prob a world where u get that
- M: q is, do we care that much about this to delay any concept of async payments til PTLCs. Probably not. Meaning we need to support the keysend+htlcs thing in the meantime
- Rusty: q is, do we have a path forward. In theory, a preimage is an arbitrary thing and could contain arbitrary data. preimage itself could be signed by the sender maybe? Handwave
- M: with HTLCs you can't change the preimage, have to change script format to commit to additional data. With PTLCs maybe could add in additional data, and in the invoice "i support PTLCs with additional data and will only claim if i know the additional data." you're committing the PTLC data that it was sent by this nonce at this time
- rusty: i dislike keysend bc of lack of proof. But if there's a path forward, I feel better. Not blocking though, let's move forward with HTLCs+keysend for async payments
- TODO(val) send out an ML email to see if someone will solve reusable-PTLC-invoices for us

Async Payments w/ HTLCs spec

- Rusty: would be good to get something spec'd for async payments. There are a number of arbitrary protocol choices that need to be made.
-

- We'll want to retransmit the "hold_htlc_avail" onion message OR have a positive acknowledgement.
- Problem with retransmission: in the current spec, the hold_htlc_avail message is sent to the final destination
- Depends whether you have the LSP respond "i am ready" or just keep retransmitting. The problem is if you get gross failure and die reaching LSP, what happens… unlikely bc you know the payment has bounced off LSP. lots of diff designs and random protocol decisions here.
- Likely you need to implement it to write the spec. But if we wanna do implementation first, we need to have finished impl for offers and everything that comes with it, so still a few months before we can do anything. blocked on offers + trampoline.
-
- Note that we'll want an explicit feature bit in offers saying "there is no payment hash here."
-
- If we really wanted, we could maybe write some demoware for async payments based on today's keysend. But "real" demoware would be blocked on offers impl

Async Payments w/o trampoline

- Rusty: talked about this originally — local LSP retransmission has some interesting things we could do, except LSP tracks you and blocks repeats (due to timing attacks, maybe), but if you could weaken that, attempt to send—
- if you try to reuse an onion, in theory the LSP could retransmit naively, but the spec says if you see the same onion, you're supposed to block it a 2nd time. stops your LSP retransmitting for you.
- If you allow retransmission 1x/hour or sth, you could have your LSP automatically retransmit for you by giving them a secret so they can decrypt the error returned, then retry. so then they set up with the other LSP, "tell me when you're ready" and retry. so it does simplify it somewhat
- Rusty: eventually you'll hit a timeout of the original invoice. we're assuming async payments go thru in some reasonable time
- rusty: i assume 24 hours
- others: i think longer
- in the case where the phone has to retransmit, idk what your timeout would be.
-
- with trampoline, lnd retransmit thing is not an issue. LSP in the original proposal, you tell it where the trampoline is and it creates a new onion. if you wanna avoid that, could do a trick where you give the LSP enough to decrypt an error from remote LSP only.

Offers tangent

- sidenote: we should interop with cln and eclair for offers
- matt: we still haven't done progress on blinded payment stuff, gonna be a while before we can land bolts from our PoV sadly

- blockstream to release currency conversion api for offers. will be nice to have an OM service to do that too
- R: we have a currency plugin, and if the api isn't avails, will give them a big warning if they issue an offer w non-btc currency.
- matt: need an onion message based oracle for currency. a few well known endpoints would be nice.

Onion messages tangent

- Ldk has OM interop with eclair and at least some with lnd
- we can merge onion messages spec w/o lnd. should prob do a scan of the spec. we're blocked on interop for payment blinded paths first sadly.
- t-bast: i can test that interop with cln soon, so that unblocks the blinded paths PR, which unblocks onion messages