This is a follow up of Time as a Public Service in Byzantine context,

Time attacks and security models and Sensor Fusion for BFT Clock Sync. Additionally, it's a “bridge” write up, which starts to discuss how Decentralized Trustless Oracles can be built, based on Sensor Fusion approach.

Thanks to Joseph Chow and my TX/RX Research colleagues for discussions.

The Clock Sync approach, we have discussed, can be viewed as an example of Decentralized Trustless Oracle (DTO). Reference clocks (for a common time standard) can be seen both as sensors or as oracles. In the clock sync approach, the reference clocks are attached to nodes, forming a distributed network of sensors/oracles. Individual sensors/oracles may fail, so a robust aggregation method is applied to fuse multiple sensors/oracles and obtain an accurate time estimates. The aggregation process can be replicated on every node, resulting in a decentralized oracle, which need not be trusted to the extent that the robust aggregation method is able to tolerate faults.

There are several topics inspired by the analogy between sensors and oracles that we'd like to explore:

- clock sync approach analysis from DTO perspective
- how the clock sync/sensor fusion paradigm can be extended to construct DTOs in general
- role of time for oracles

The write up concentrates on the first one. The rest are to be discussed in following write ups.

Clock Sync as an External Time Oracle

Synchronous distributed protocols or protocols operating in a lock-step manner need a clock sync protocol (e.g.Abraham et al. [2017], section 5). Note however that such protocols don't necessarily need a common time standard (and thus do not need reference clocks plus) - as they can synchronize (and calibrate, if needed) their local clocks. However, anchoring to a common time standard is beneficial from cryptoeconomic (participants rewards can depend on time) or from application perspectives (e.g. smart contracts need time service), more discussion is here.

We may thus make an accent on the oracle facet of the clock sync approach: it's a DTO (sensing a common time standard), which can be used as a clock synchronization mechanism (required by the corresponding synchronous or lock-step distributed protocol).

Such accent entails some consequences (we'll discuss further some of them). First, the oracle provides input on the protocol level, not on the smart contract level (as is typically assumed in oracle literature). Second, individual sensors/oracles are collocated with nodes participating in a blockchain agreement protocol. Which is somewhat unusual too (though it hardly can be called unique). While it is not a scalable solution (considering oracles for other kinds of external data), it brings certain benefits: oracle agreement is built in parallel with a blockchain agreement, so common infrastructure can be shared and operating costs can be reduced (e.g. clock sync can piggyback blockchain protocol message flow).

Additionally, we'd like to note that in a distributed setting, “external data” assumes some “common standard”: to be able to aggregate a set of (distributed) sensors - and build a single (decentralized) sensor - the sensors should be sensing the same substance.

In the case of time oracles, sensing time takes time itself: there are inevitable delays when communicating information. However, each kind of event can be associated with a timestamp (or even several kinds of them). In a result, the uncertainty of measuring time can spill over other kind of oracles. As we (as well as the NTP standard) employ interval data in the aforementioned clock sync, the interval approach can be extended to model other kind of oracles.

Definition of Oracle

As we employ time oracle in a somewhat different manner, we need to provide an explicit Oracle definition that suits our needs. A typical Oracle definition assumes that oracle provides external data on a smart contract or on a blockchain level. Our update to such a definition is not dramatic: our oracle may operate on a protocol level. We thus use the following phrase:

Oracle is an entity providing external data to a BFT solution.

Sometimes, an Oracle definition assumes that an oracle signs its output. We avoid it, since while cryptography signatures help to build BFT protocols, they are not strictly necessary. So, we consider signatures as a part of wider set of means to assure trustlessness.

Trusted vs Trustless Oracles

An oracle may fail for various reasons. Therefore, any BFT solution relying on such an oracle, may become compromised.

One has either to add additional assumptions, i.e. put the oracle in a list of trusted components - Trusted Oracle. Or to use an implementation, which is able to tolerate faults.

Centralized vs Decentralized Oracles

A centralized service may trivially fail as it's a Single Point Of Failure (SPOF). We therefore define decentralized oracle as an oracle without SPOFs. Though, in practice, it can be impossible to eliminate every one and each SPOF.

For example, let's consider time oracles, which provide access to some common time standard, e.g. UTC. The standard itself or services related to it can be considered as a SPOFs. In theory, many independent extremely stable (e.g. atomic) clocks can be synchronized to UTC and keep approximate UTC time for a long period. Though, sooner or later, the accrued discrepancy will become too large and such set up can be expensive.

Therefore, in practice, one has to trust the time standard and its core distribution infrastructure (e.g. GPS/GLONASS/radio clocks, etc).

So, decentralized

is somewhat vague property, however, one can define a set of assumptions about failures that should be tolerated.

Clock Sync as a DTO

We assume that a blockchain protocol relies on a clock sync protocol, but the clock sync approach BFT properties rely on certain assumptions too. The clock sync for a beacon chain protocol approach partially relies on the beacon chain protocol. In the section, we analyse them, based on a set of criteria inspired by Decentralised Oracles: a comprehensive overview.

"Trustlessness" properties

Truslessness is not a black-or-white concept, in fact, there can be many aspects why someone can (dis)trust something. We consider the following set of properties aspects:

- consensus BFT properties
- incentives (cryptoeconomic)
- sybil attack resistance
- free rider resistance
- confidentiality/privacy concerns

Concensus BFT properties

The clock sync approach is based on a variant of inexact consensus, which usesMarzullo's or Brooks-Iyengar algorithm, which output a correct interval if majority of input intervals are correct.

However, in practice as stricter assumptions may be necessary as the clock sync relies on the beacon chain protocol to counteract other risks (e.g. Sybil-attacks).

Precision may also benefit from stricter assumptions.

Incentives (cryptoeconomic aspects)

What are incentives for an individual oracle to report true value to other participants?

We are buiding on beacon chain protocol too here: as the clock sync piggyback beacon chain messages, reporting wrong values mean messages either earlier or late, which is penalized by beacon chain protocol (if a message is too off).

Collusion can reverse the situation, however, we have to expect that majority is well behaving so that robust aggregation work properly.

Sybil attack counter-measures

An adversary can (relatively) easily obtain majority if it doesn't cost much to add a participant. So, any practical BFT solution has to restrict the set of participants either with economical or administrative restrictions: PoW, PoS, PoA, etc.

As clock sync's DTO is collocated with main protocol nodes, it's also natural to counteract Sybil-attacks, but requiring that only main protocol participants can be time DTO operators. Cryptographic signatures can be used to enforce it - it's also natural if the time DTO info piggybacks main message flow.

Free riders/freeloading

A participant can evade costs associated with a proper reference clock set up, which is hardly possible to detect. Such participant can live without any reference clocks, as it is receiving messages from other participants and so can extract time info from them.

Such behavior doesn't affect clock synchronization between participants, however, the overall system relies on less reference clocks, than expected. That means it's easier for an adversary to shift resulting DTO clock relative to true value (i.e. common time standard).

Some DTO approaches use commit-reveal protocols to counteract freerding (see e.g. here), however, it's hardly possible in the case of time oracle, as NTP clock synchronization is easy to set up (and often it's set up by default). So, some reference clock is available, but the problem is that it can be vulnerable, as NTP infrastructure is relatively centralized and/or relatively easy to manipulate.

While better reference clock options are available and not always expensive (e.g. GPS clocks), it still requires some efforts and/or expertise to properly set up. And such options can be much more difficult in the case of hosted nodes.

Consider a case, when only a small percentage of nodes has proper reference clock setups and others are either free-riding on the exisiting message flow or are nodes, controlled by an adversary. Additionally, let's assume that the adversarial power is enough to outweigh the small percentage of honest nodes. Then the adversary can shift consensus clocks relative to the world standard. It's possible for an external observer to detect the shift. But how this can be translated into actions? Obviously, one more time DTO is needed.

Privacy concerns

There is hardly any privacy concerns with time data

UPD @hoytech noted below about some concerns.

Potential ways to counteract Free-riding

Free-riding seems to be the most important problem in the case of Time DTO. A tradtional approach is commit-reveal protocol, however, in the case of Time Oracles, there is nothing to hide - e.g. a node can easily use an NTP server as a reference clock.

The problem is that NTP is not designed to be fault tolerant to the level, which is required for blockchain solutions. So, there is either a centralization risk (as some popular NTP servers are operated by corporations) or a Sybil attack risk (as NTP pool is free to join).

There can be several approaches, which can help to alleviate the freerider problem, but it's not clear whether it can be completely resolved, assuming rational Time Oracle operators. We briefly cover them, however, a more detailed and rigorous discussion is delayed until further write ups.

One approach is to build Time DTO using PoA approach, i.e. Time DTO operators can be incentivesed to use proper reference clock setups with a reputation risk. Additionally, institutions have more resources to do that, compared to 'retail' validators in the permissionless setting.

Another approach can be a hardware solution, e.g. combining Trusted Execution Environment with a proper reference clocks (e.g. an atomic clock). Such hardware can sign its output which can be verified by others.

A third approach can be to 'discipline' a Time DTO exposed to the freerider risk with an additional 'governance' DTO. Since another DTO is required, it may looks like no solution at all. However, such DTO can operate in a different time frame. If there is an attack shifting output of the first DTO relative to the common time standard, then it will be noticed by humans. So, human governors can vote to slash or to penalize Time DTO operators. This creates an incentives for Time DTO operators to use a proper clock setup.

In our opinion, PoA Time DTO is the most realistic approach, for example, in a permissioned blockchain setting.

Conclusion

Sensor Fusion/Clock Sync approach can be seen as a Time DTO. There are problems (like cryptoeconomic incentives and Sybil-attacks countermeasures), which can be resolved, if such Time DTO is combined with a blockchain protocol.

There are two problems which are more difficult to solve. First, a common world time standard (like UTC) should be trusted - it's hard to avoid.

Another problem is free-riding, in the case of Time DTO, it is a proper reference clock setup, which can be costly or inconvinient (e.g. in a permissionless setting).