TL;DR

Presenting a.DI

(Aave Delivery Infrastructure

), a cross-chain communication abstraction layer for a decentralized system like the Aave DAO to communicate across networks, minimizing the risk of any underlying bridge providers.

Context

Aave, a cross-chain ecosystem

Starting on Ethereum with v1, through v2 and v3, Aave became a system living across different networks.

Even if the instances of the liquidity protocol are completely isolated, the control over them lives on Ethereum, where the decentralized Aave Governance and its voting assets (AAVE, stkAAVE) reside.

So a communication layer of governance decisions was a must.

Limitations of the current Aave cross-chain governance

From the first Aave v2 deployment in a non-Ethereum network (Polygon), an Aave cross-chain governance system was introduced, allowing the Aave Governance on Ethereum to control those instances by bridging decisions.

However, even if highly innovative, the system had certain limitations, mainly caused by the incredibly early stage of bridging infrastructure years ago. These constraints are:

- Total dependency on a single underlying bridge.

The current cross-chain governance system requires “choosing” a single “bridge provider”, and once done and connected, the reliance on it is total.

This is the reason why only native/official bridges have been accepted, as technical details apart, they are the most heavily tied entities with the network itself they connect to (e.g. Polygon bridge).

- Friction when trying to expand to new networks.

The consequence of the previous point, when expanding Aave to new networks, fully on-chain decentralized governance is only possible whenever there is a native/official bridge available, which is not always the case (e.g. Avalanche)

State of the art on current bridging technologies

The landscape of bridging technologies has radically changed during the last 2 years. While before the technologies ready for production usage were really scarce (mainly native/official bridges of networks), now there are multiple players with sound infrastructure and many more continuously developing products in the field.

Even if security is still a pretty big concern in cross-chain communication, for protocol/blockchain-application layers like Aave, this proliferation of solutions is a clear indication that something can be improved, starting with Aave cross-chain governance.

Relation with Aave Governance v3

As described in the announcement post of Aave Governance v3, a robust cross-chain infrastructure is an important dependency, given the high level of cross-chain communication included in the governance flow.

So even if a.DI is a standalone system, it is a fundamental component of Aave Governance v3.

What is a.DI? How does it work?

a.DI (Aave Delivery Infrastructure) is a cross-chain abstraction layer of smart contracts, without any off-chain component, focused on giving full sovereignty to the Aave DAO with minimal trust assumptions on underlying bridge technologies.

The design principles of a.DI are:

- DAO first, but generic.

a.DI has been designed for the Aave DAO by community members that deeply understand its needs. So the focus has been covering its present and future needs.

But as a side-effect to be future-proof, the system is also quite generic one that can be adapted for almost any kind of cross-chain communication needs.

- Full control, no trust.

a.DI reduces the trust in the underlying used bridges to the minimum. This means that the entity controlling a.DI (Aave governance) can replace any underlying bridge if malicious/exploited, define consensus rules between multiple bridges, or even survive an emergency event on which no bridge is functional.

- Security abstraction; consensus-based.

Securing bridges is a pretty difficult challenge, with numerous exploits during the last few years. In addition, even doing a deep evaluation of their security model is really complex work, given how different they are from each other and their granularity of components.

The approach with a.DI changes this: being based on consensus and with recovery mechanisms, by adding minimally trusted bridging providers to the set, the security improves. This is especially applicable to “slow” systems like the Aave governance.

- Good development experience and well-defined flows.

Operational overhead is an important problem for DAOs, so simple-enough developer experience and good guidelines for contributors are fundamental. The abstraction provided by a.DI to cover the other principles helps with this too as a side effect.

The components of a.DI

Communication flow

On a message communication protocol, everything starts with a message sent from an address and ends with a message delivered to another address.

But internally, the system works as follows:

Sending phase

1. A user (e.g. the Aave governance) sends a string of bytes (message) to the Cross-chain Controller smart contract (CCC), the entry point of a.DI.

2. The CCC wraps the message internally via its Cross-chain Forwarder component (CCF) into a transaction: an internal format of a.DI, not to confuse it with the blockchain concept of a transaction.

3. The CCF sends the transaction through each one of the authorized bridge providers, transparently handling the way of interacting with each one of them or paying bridging fees.

Receiving phase

1. Each one of the authorized bridges sends the a.DI transaction to the CCC on the receiving network via adapter smart contracts.

2. Within the CCC, the Cross-chain Receiver (CCR) component applies consensus rules. E.g. if communication Ethereum → Polygon requires 2-of-3 consensus on authorized bridges, only when 2 out of them deliver equivalent transactions on the CCR, this will be forwarded to the final destination.

3. After all the unwrapping from internal formats of a.DI and with consensus reached, the final message is forwarded to the destination address.

Something to highlight is that a.DI is direction agnostic.

The previous flow applies to any pair of networks connected: Ethereum → Polygon, Polygon → Ethereum, Ethereum → Avalanche, or any other.

Additionally, it is possible to abstract same-chain communication via a.DI, something important in systems like Aave Governance v3 (Ethereum → Ethereum).

Control model. Operational vs Emergency modes

Even including mechanisms like consensus over multiple bridge providers, there can be cases where they are not enough for governance like Aave to keep full control of its permissions. For example, an edge scenario on multiple bridges malfunctioning simultaneously could create a “blockage” on governance, not being able to bridge decisions and replace them with working ones.

At the same time, adding a third party (like the Aave Guardian) with full power to replace bridges would make the system meaningless and, factually, would give this third party full control, even if not exercising it.

The solution included in a.DI is the distinction between 2 working modes of the system: operational and emergency.

- Operational mode.

This is the “standard” mode of aDI, on which full control over management actions of aDI on the Aave governance: for any change of configuration (e.g. authorizing or disabling a bridge provider, change consensus rules), a governance proposal needs to be passed, that will send a message via a.DI, to finally change its parameters.

No other entity apart from the Aave governance has any control in Operational mode

.

- Emergency mode.

Now let’s assume a practical example of the previous “blockage” scenario.

Let’s say Ethereum → Polygon on a.DI has bridges A, B, and C as authorized and a 2-of-3 consensus. Also, a.DI is configured as the only party that can pass messages to a contract controlling both a.DI on Polygon and Aave v3 Polygon.

For some edge reason, both B and C stop operating and passing messages, which means that the Aave Governance is incapable of passing a message via a.DI to the contract controlling both a.DI and v3 Polygon, locking both.

To avoid that situation, the a.DI includes a mechanism that works this way:

- The Aave Governance on Ethereum has control over a contract on Ethereum, on which it can signal the activation of an emergency event for a specific network id, by passing a standard governance proposal.
- If an emergency event is activated on Ethereum, we have developed a system together with Chainlink (completely independent from a.DI itself) which ad-hoc will detect it and set a flag on the signaled network.
- Whenever that flag gets enabled on the affected network, a.DI there will detect it and will automatically change its own access control model: a Guardian entity will get permissions to execute the so-called solveEmergency()

action, factually getting full power to replace bridges or changing other configurations. Afterward, permissions are removed from the Guardian, and to activate emergency mode again, a new emergency event should be raised on Ethereum.

- The Aave Governance on Ethereum has control over a contract on Ethereum, on which it can signal the activation of an emergency event for a specific network id, by passing a standard governance proposal.
- If an emergency event is activated on Ethereum, we have developed a system together with Chainlink (completely independent from a.DI itself) which ad-hoc will detect it and set a flag on the signaled network.
- Whenever that flag gets enabled on the affected network, a.DI there will detect it and will automatically change its own access control model: a Guardian entity will get permissions to execute the so-called solveEmergency()

action, factually getting full power to replace bridges or changing other configurations. Afterward, permissions are removed from the Guardian, and to activate emergency mode again, a new emergency event should be raised on Ethereum.

This system follows a similar approach as the Price Oracle Sentinel on Aave v3 and allows for a pretty powerful outcome: while in operational mode, a.DI has no third party with any control apart from the Aave governance itself. But if the Aave governance decides so (and only in that case), an extra entity like the Aave Guardian can gain temporary power to solve any problem

, not possible otherwise.

The following diagrams show the difference between operational and emergency modes.

Configuration of underlying bridges

In parallel with the design of a.DI, we have been evaluating and integrating into the system different underlying providers, as even if the community will be able to re-configure this in the future, we firmly believe the first proposal should come from our side.

The following aspects have been considered during this integration/selection effort:

- Multiple bridges only when there are security advantages

. Depending on the morphology of networks, in some cases, the native/official bridge shares exactly the same security as the network itself (e.g. Optimism), while in other cases, the network and bridge are technically separated entities, with divergent control models (e.g. Avalanche, with no native/official bridge or Polygon PoS).

- Proved technology with no major security concerns

. A minimum maturity in the market in the product and/or team maintenance was a consideration for us.

- Minimal non-intersection between participants on the bridge's internal network, between bridges

. By general rule, all bridge providers are based on some type of network of entities participating in the system, usually in monitoring and relayer tasks.

As a.DI is based on consensus, it is especially important that those participants on Bridge provider A do not completely overlap with other Bridge providers in the consensus set.

- Meaningful architectural attempts on the bridges to avoid centralization or plan to

. Centralization is usually a big consideration in the evaluation of bridges, but a.DI is specifically designed to be resilient against centralization. This directly translates to our evaluation of candidates regarding centralization: all bridge providers should be making meaningful attempts to decentralize their service. But if there are relatively centralized components, if this doesn't critically compromise security, we consider it acceptable and prioritize having a sound infrastructure in for example uptime.

At the moment, we can confirm the following bridges will be part of the initial list:

- LayerZero
- CCIP
- Hyperlane
- All native/official bridges of networks where Aave is present

In parallel, we are evaluating and/or integrating extra candidates, which depending on the timeline will be included at launch or afterward.

FAQ

What is an acceptable consensus on the recipient side of a network?

It depends. A 2-of-3 generally is a good starting point, and scaling that proportion can be considered if extra reliable options exist.

Is there any mechanism to "veto" messages?

No. Consensus on a.DI determines what is correct and not: in a 2-of-3 consensus, if 2 bridges agree, the message is correct. The only way to "pause" the system would be to raise an emergency state from Ethereum.

Systems on top should implement extra protections. For example, Aave Governance v3 incorporates a veto mechanism by Guardian, with timelock.

What are the running costs of the system for the DAO?

The cost depends on the bridge providers used and their pricing mechanics. In addition, there is some extra gas overhead of the a.DI layer itself.

As cross-chain communication on the Aave Governance (both v2 and v3) is almost always fixed in size in bytes, the cost of messaging will be pretty low in practice, in the order of 1-5 dollars per message.

Can a.DI be used for other needs of cross-chain communication of the DAO, apart from Governance?

Yes, the system is generic. Depending on the case, maybe a new instance of a.DI needs to be deployed, to have different consensus rules and bridge providers, but that should not be an obstacle.

Who can send messages via a.DI?

The initial instance of a.DI will be whitelisted for Aave Governance smart contracts. This is because the system itself will pay its cost with funds of the DAO, so only messages that the DAO is willing to cover the cost for should be allowed.

What happens if a message is sent and there is any problem with bridge providers in the middle? Will the sender need to send it again?

Generally, the answer is no. a.DI includes a retry mechanism on top of the retry mechanisms of the underlying bridges to avoid situations like the Aave Governance sending a message → message delivery not going through due to underlying bridges → governance proposal needs to be re-submitted.

Activation steps & Timeline

As previously mentioned, a.DI is heavily linked as a requirement to Aave Governance v3, with the activation of both being simultaneous in production.

After this post, the triad of Aave Governance v3 related components have been presented to the community: Aave Governance v3, a.DI and Aave Robot (in what concerns permissionless automation of the two previous systems).

The timeline and missing steps until activation are:

1. In parallel with this post, we will create thepreviously announced Snapshot for the community to initially approve the Aave Governance v3 triad of components.

The requirements will mirror Level 1 on-chain proposals: minimum 320’000 AAVE Yes votes and 80’000 differential between Yes and No.

1. All security procedures are getting finished, including the ones of Certora and SigmaPrime, both already well advanced. Once that happens, we will release the final codebase.
2. We will finish the preparations of the whole setup, including all the deployments and the final payload for AIP that will activate all the systems.

This procedure will be worth it for a post by itself, which we will publish once ready.