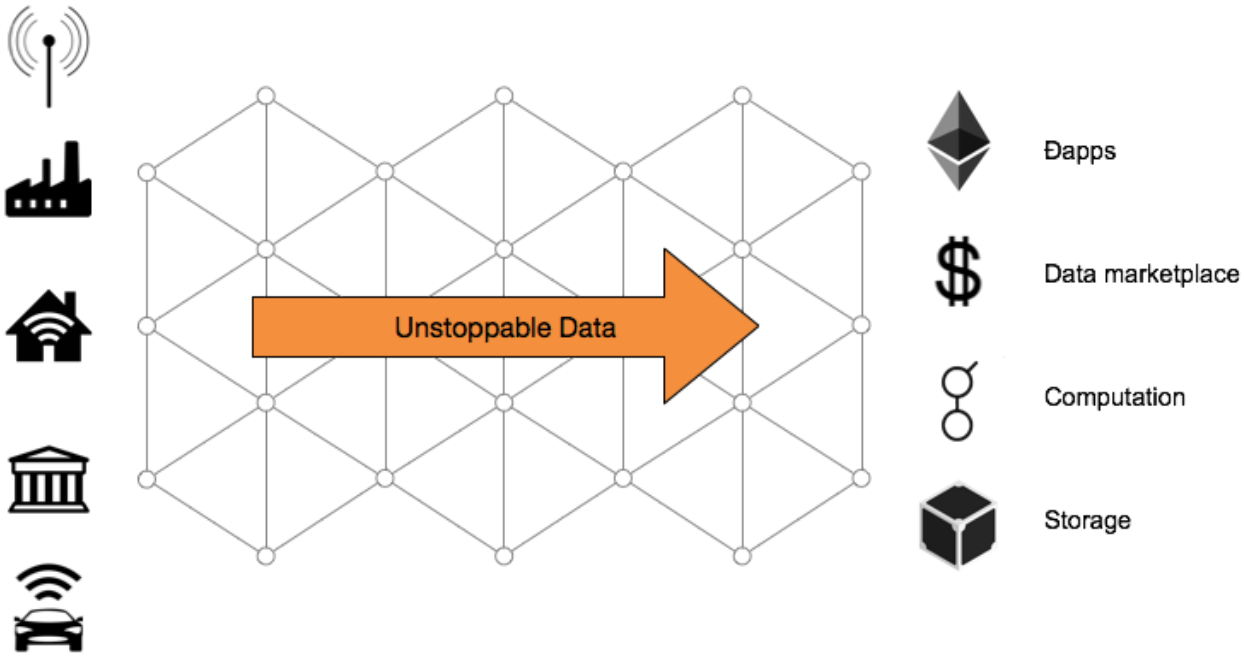
Unstoppable Data for Unstoppable Apps: DATAcoin by Streamr

streamr

July 25th, 2017
Version 1.0

This whitepaper is for information only and does not constitute an offer or any kind of investment advice. Any element of this whitepaper may undergo significant changes as the project further develops.



Streamr vision

Streamr delivers unstoppable data to unstoppable applications. It is the real-time data backbone of the global supercomputer. It is a decentralized network for scalable, low-latency, untamperable data delivery and persistence, operated by the DATAcoin token. Anyone — or anything — can publish new data to data streams, and others can subscribe to these streams to power Đapps, smart contracts, microservices, and intelligent data pipelines.

To incentivize user participation in the network, there's a built-in mechanism for data monetization. Valuable data from security exchanges, connected devices, IoT sensors, and social media can be offered to companies, developers, and private citizens. Machines can autonomously sell their data, get paid, and purchase the data they require. A global market for real-time data emerges, with built-in data provenance, encryption, and access control.

Alongside the decentralized data network and marketplace, the full Streamr stack includes a powerful analytics engine and a UI for rapid development of real-time Đapps. Data streams, smart contracts, and decentralized computing resources can be interconnected in a low-code environment using high-level building blocks. Streamr will be the easiest place to create real-time, data-driven, and trustworthy blockchain applications.

A revolution is taking place where centralized cloud services are one by one being superseded by tokenized, decentralized solutions. Golem, for example, replaces Azure Virtual Machine, and IPFS replaces Azure Blob Storage. Streamr is proud to join the revolution by providing a decentralized solution to messaging and event processing, replacing platforms such as Azure EventHub and Azure Stream Analytics.

1. Background

Real-time data will increasingly turn into a commodity in the coming years. Huge volumes of timestamped data is being generated by sensors and connected devices in manufacturing, the service sector, and the entire supply chain which underlies the modern economy, with much of the data generated in a streaming fashion[1,2].

The amount of data is increasing exponentially along the growth of IoT and the ubiquity of connected devices. In the global IoT market, IHS Markit forecasts[3] that the installed base will grow from 15.4 billion devices in 2015 to 30.7 billion devices in 2020 and to 75.4 billion in 2025. Much of the newly generated data is valuable: It can be used to optimise manufacturing operations, track assets with increasing accuracy, target existing consumer services with high granularity, and create entirely new services and business models.

At the same time, there is a megatrend in motion towards the next generation of the computing stack. In a distributed future, the backend code of decentralized apps — or Đapps[4] — runs in peer-to-peer networks. Ethereum is a Đapp in itself, so is Golem, and there are many more in development.

However, Đapps do not run in isolation: They need external data to function. As it is, storage and distribution of real-world data remain centralised, and Đapps remain liable to all the known problems: Concentration of power, lack of robustness, and vulnerability to cyber attacks.

To be sure, you can already store data in the blockchain. There are also decentralized file storage apps such as IPFS, Swarm, and Storj, and databases like BigchainDB are starting to emerge. While such solutions are surely part of the new decentralized fabric, they don't really provide an answer to cases where real-time data is needed in any significant volumes. The chain is not designed for high throughput or low latency, it does not scale, and storage is expensive.

What is needed is a natively decentralized data backbone as a complement to decentralized apps. This real-time data backbone will be the missing link, and the link that we want to help provide. The infrastructure we create consists of a technology stack which helps connect and

---

[1] Susan O'Brien: "5 Big Data Trends Shaping the Future of Data-Driven Businesses", Datameer, 11 May 2016 (https://www.datameer.com/company/datameer-blog/5-big-data-trends-shaping-future-data-driven-businesses/)
[2] Tony Baer: "2017 Trends to Watch: Big Data", Ovum, 21 November 2016 (https://ovum.informa.com/~/media/Informa-Shop-Window/TMT/Files/Whitepapers/2017_Trends_to_Watch_Big_Data.pdf)
[3] Sam Lucero: "IoT Platforms: enabling the Internet of Things", IHS Markit, March 2016 (https://cdn.ihs.com/www/pdf/enabling-IOT.pdf)
[4] For a definition of Đapps, see Johnston et al.: The General Theory of Decentralized Applications (https://github.com/DavidJohnstonCEO/DecentralizedApplications)

incentivise computers in a global peer-to-peer (P2P) network. This is a network which provides low-latency, robust and secure data delivery and persistency, and all at scale. Đapps of the future are fuelled by data, and our mission is to make sure that the data keeps on flowing.

We also create a market for real-time data. In the data market, anyone can publish events to data streams, and anyone can subscribe to streams and use the data in decentralized apps. Much of the data is free, but where that's not the case, the terms of use are stored in Ethereum smart contracts. A digital token — a DATAcoin — is needed to access and operate the data market, and to compensate nodes in the P2P network. Subscribers pay for the data with the token, and data producers and network participants are reimbursed automatically and securely.

Our stack is built on a decentralized transport layer. Apart from greater robustness, resilience and fault tolerance, decentralization facilitates openness, transparency, and community building. The power over data is not with large corporations like Google, Amazon, Microsoft, and IBM. The network consists of a multitude of data producers, data consumers, and message broker nodes in-between. You make a reputation for yourself and earn good karma by contributing to data exchange and by helping run the network to everyone's benefit.

We believe that sustained growth of the blockchain community will be facilitated by having a good usability layer in place. Tools are needed so that non-experts can create secure smart contracts, and connect those contracts and Đapps to reliable data sources. We will help build the required toolkit by providing a visual editor, wrappers, and templates. In short, we want to be the place to go for anyone who's in the business of creating data-driven decentralized services.

In the rest of this paper we describe the Streamr technology stack, define the role of the digital token, explain the status quo, present the R&D roadmap, and introduce the team.

2. Streamr stack

The decentralized real-time data pipeline is built on top of a multi-layered technology stack:

- **Streamr Editor** constitutes a usability layer and toolkit which enables rapid development of decentralized, data-driven apps.
- **Streamr Engine** is a high-performance event processing and analytics engine that executes off-chain in a decentralized fashion. It can run on a decentralized computing provider such as Golem.
- **Streamr Data Market** is a universe of shared data streams which anyone can contribute and subscribe to.
- **Streamr Network** is the data transport layer, defining an incentivized peer-to-peer network for messaging in the decentralized data pipeline.

- **Streamr Smart Contracts** enable nodes in the Streamr network to reach consensus, hold stream metadata, handle permissioning and integrity checking, and facilitate secure token transfers.

The following section goes through each layer of the stack (see Figure 1) in detail, following a top-down approach.

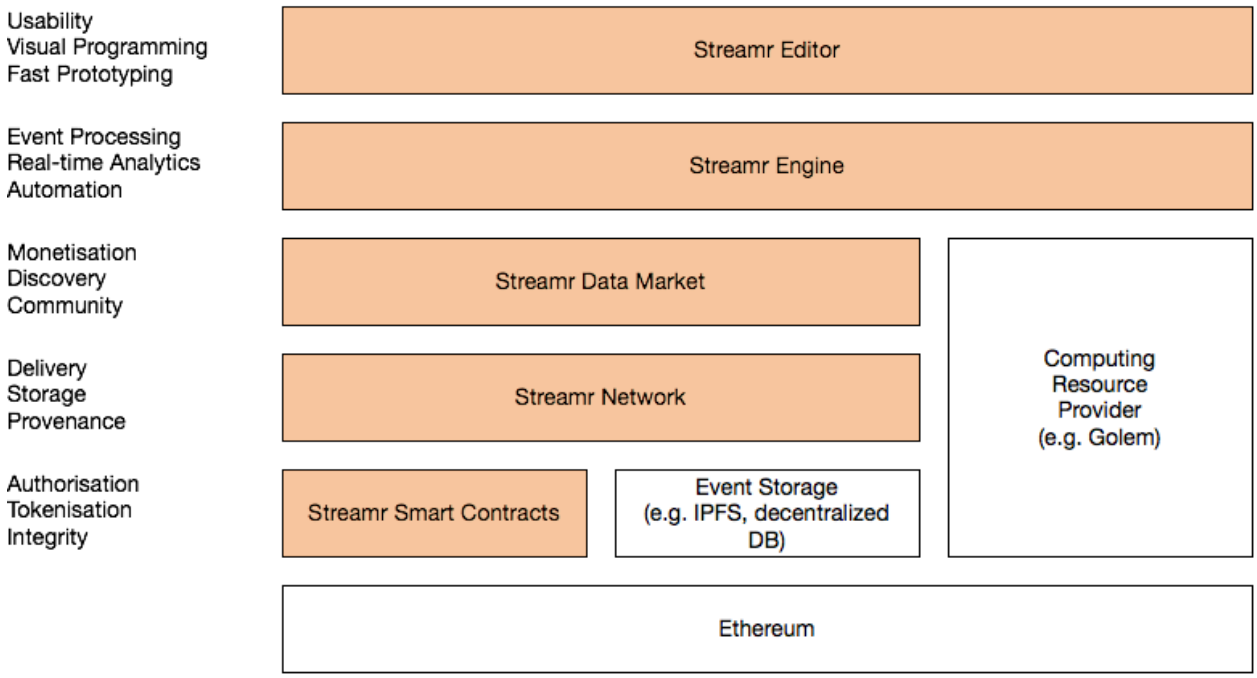


Figure 1. Streamr technology stack.

2.1 Streamr Editor

> Streamr Editor enables rapid development of data-driven smart contracts, lowers the threshold for Đapp creation, and comes with ready-made templates for common use cases built in.

There is considerable interest in the blockchain and decentralized applications within the business community, but the number of real-life use cases remains limited. These are the early days, and it is not unreasonable to postulate that many of those who want to get involved are not deep experts in the minutiae of Ethereum, Solidity, encryption, data provenance, and other technical issues.

In our view, the commercial growth of the ecosystem requires tools which allow non-experts to set up smart contracts, connect to trusted data sources, make use of secure off-chain modules for data filtering, aggregation, and refinement, deploy decentralized applications, track smart contract execution, and visualise the flow of input data and blockchain events.

We address the need for a usability layer by providing powerful tools (such as an easy-to-use visual editor), wrappers, and smart contract templates which are targeted at domain professionals and business users. These tools hide the deep technology under the hood, handle the data integrations and communications, and automate the routine steps in the deployment and monitoring of smart contracts.

We foresee an ecosystem where there are several usability platforms and tools available. The existing Streamr platform already implements some elements of the usability layer, with more functionality being added in the coming months and years. The aim is to reach a stage where you can build and deploy a useful and functioning data-driven smart contract in minutes. This is more than a fantasy; our demo in EDCON[5] Paris in February 2017 is a taster of what can already be done (see Figure 2 for an illustration).

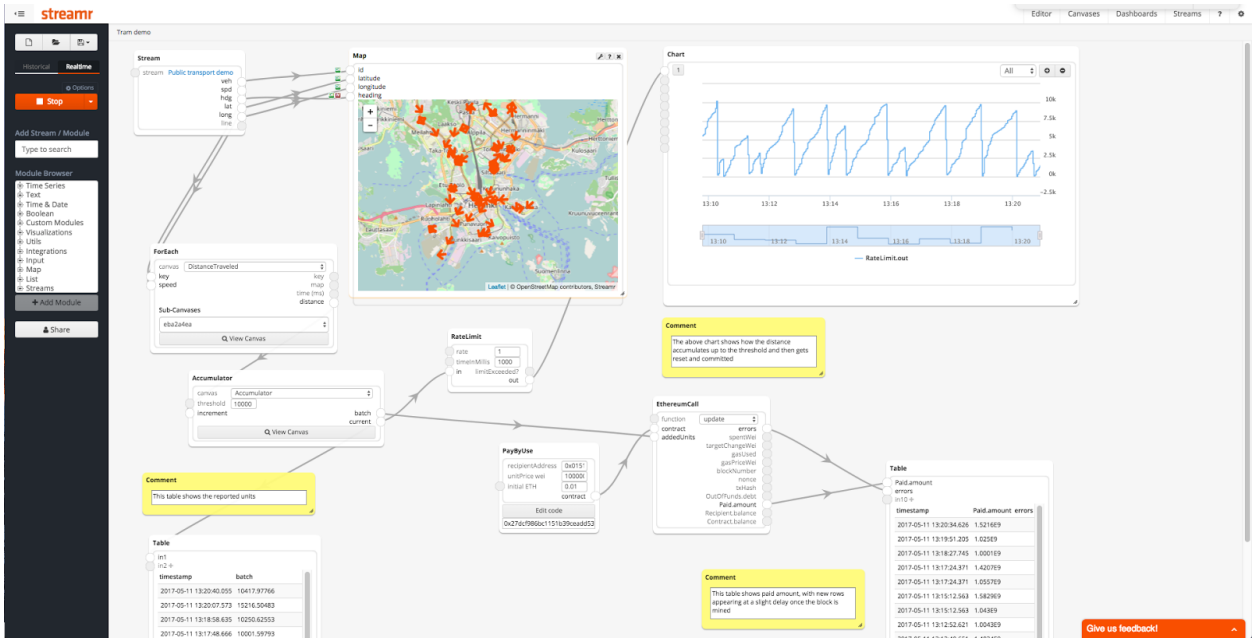


Figure 2. An alpha version of the Streamr editor workspace.

These are some of the planned features for the usability layer:

[5] Henri Pihkala: "Connecting Ethereum with the real world: How to easily create data-driven smart contracts", European Ethereum Development Conference (EDCON), Paris, 17-18 February, 2017 (https://www.youtube.com/watch?v=C1l0rcj-Fok)

- A visual editor for creating smart contracts, feeding in real-world data, and constructing off-chain data processing pipelines.
- Modules for communication with smart contracts and interacting with the blockchain.
- Modules for off-chain processing: Data filtering, refinement, and aggregation, deployment of decentralized applications, tracking smart contract execution, and visualising the flow of input data and blockchain events.
- A Solidity editor where the smart contract code can be written and modified in a context-sensitive environment.
- Built-in and tested open source Solidity templates for different use cases of Ethereum smart contracts.
- Playback functionality for simulating smart contract functionality, debugging contract code, and testing functionality before deployment.
- One-click deployment to submit a smart contract in either a testnet or in the mainnet.

2.2 Streamr Engine

Streamr Engine is the high-performance analytics engine that executes off-chain within a decentralized computing provider (e.g. in a Docker container on Golem).

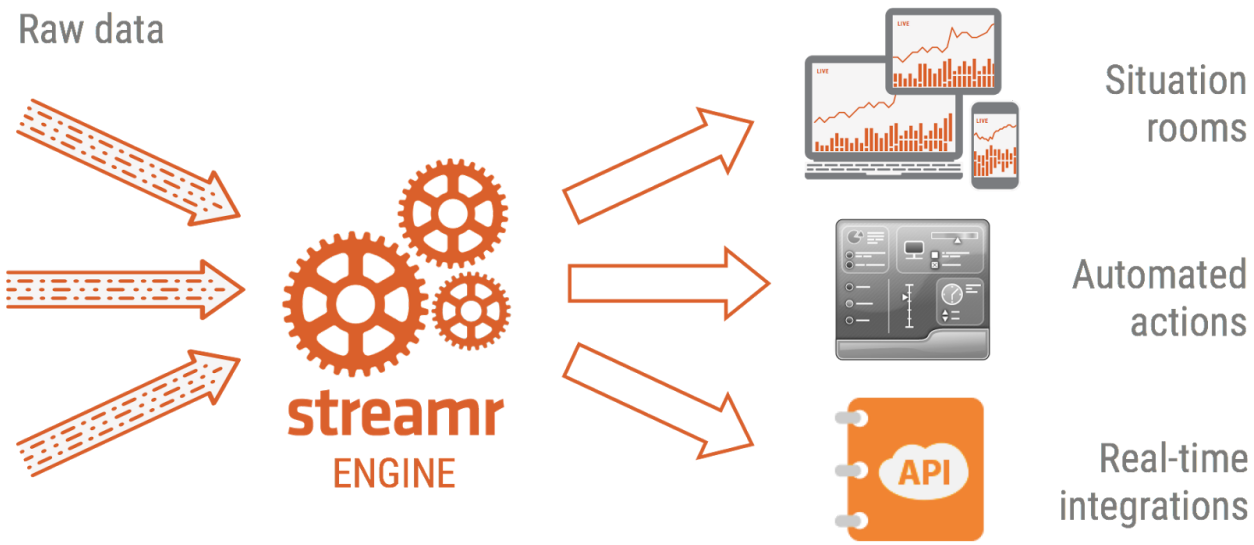


Figure 3. Typical data flow pattern and outcomes for the Streamr analytics engine.

Đapps, usually with web-based UIs and smart contract-based back-ends, currently have no way to process raw data and turn it into information. A group of IoT sensors or the stock market might produce thousands or even millions of events per second, a quantity impossible or far too expensive to push onto any blockchain for computation.

A streaming analytics layer is needed to turn raw data into refined information and ready for consumption by Đapps and smart contracts. Raw data may need to be filtered, downsampled, aggregated, combined with other data, run through anomaly detection algorithms, or processed by advanced machine learning and pattern recognition models. Or you may want to do things which simply cannot be done in smart contracts, such as calling external APIs as part of the processing chain.

The Streamr Engine listens to events on the Streamr Network, and models built using the Streamr Editor refine incoming data and react to new events in real time. There are many ways to react, including the following:

- Publishing refined data in another stream in the Streamr Network, perhaps shown in real-time by a Đapp UI also connected to the network.
- Interacting with an IoT device, for example controlling an actuator, opening a lock, turning the lights on, or calling the elevator.
- Sending an alert via email or push notification.
- Calling a function in a smart contract.

Using the Streamr Network as messaging glue between Đapps and off-chain computation on the Engine enables a whole new category of decentralized apps: apps driven by non-trivial data volume. Obviously, the results can also be consumed by traditional centralized apps, while still enjoying the benefits of decentralized messaging and analytics.

2.3 Data Market

> Streamr data market is a global universe of shared data streams which anyone can contribute and subscribe to. It's a place for data monetisation and machine-to-machine (M2M) data exchange. The data market supports anonymity, but allows for the verification of digital identity where required.

The data market is a meeting place for data producers and data consumers. Data consumers find value in the data offered, and wish to access it in order to use it as input for Đapps, smart contracts, or traditional apps.

The data is organised in data streams, the basic building block of the data market and a primitive in the Streamr Network (see Chapter 2.4 below). Data streams contain events from data sources that keep emitting new data points at either regular or irregular intervals. Here are some typical settings where real-time data is produced in a streaming fashion:

- A stock market generates a new event every time there is a new bid or offer, and every time a trade takes place.
- A public transport vehicle broadcasts its identity, status, speed, acceleration, geolocation, and heading every few seconds.
- A motion detector transmits a signal when a moving object is detected in its range.
- IIoT sensors attached to an electrical drive measure the temperature, speed, and vibrations during the drive operation in a smart factory.
- Air quality sensors measure carbon monoxide, sulfur dioxide, nitrogen dioxide, and ozone levels in an urban area.
- Seismometers measure ground motion in an area with volcanic activity.
- Smart clothing worn by professional athletes collects biometric data such as the heartbeat, temperature, and acceleration.

The data market makes a wide selection of trustworthy timestamped data available for subscription. Some of the data is sourced from established and professional data vendors and redistributors, and some from public, open data sources. Importantly, the platform allows anyone to contribute and monetize their data. Whilst companies have valuable data streaming in from sensors and devices, private citizens are producing valuable information too.

For example, people wearing a smartwatch might place their heart rate data on sale on the data market. Data can be offered anonymously, so privacy is not violated. Who would be interested in such data? Well, a pharmaceutical company might buy it for research, or a public healthcare organization might use it to find out how often people do sports, or what the stress level of the public is. A smartwatch manufacturer might buy it to get diagnostics on how their heart rate sensors perform. And the data producers earn daily income just by making their data available.

There is no reason why subscriptions in the data market should be initiated by human software developers, data engineers, or data scientists. In fact, the decentralized market may well end up being dominated by machine-to-machine transactions. Autonomous machines, robots, smart appliances will all need data in their operations, and they are producing data which is valuable to other participants in the ecosystem.

Automatic, value-adding refinement patterns will emerge. An AI might subscribe to a raw stock market feed, apply proprietary pattern recognition to generate trading signals, and offer those signals for sale on the same data market.

Whilst much of the content in the data market will be freely available for all, there will be data that needs to be paid for, and there will be data where an end user license applies. In such cases, a subscription license is needed. A license gives the right to access the data for a specific period of time, on certain conditions, and for a fee. There's a close analogy to streaming music: You don't get to own the subscribed data, any more than you get to own the rights to a song by hearing it on Spotify or by downloading it from iTunes.

Data licenses are implemented as smart contracts (see Section 2.5.4). The great benefit of the blockchain is that it offers a trustless and decentralized way to store the terms of use and the access rights, and to ensure that data payments are made as agreed.

In a wider context, there's potential for a powerful network effect in the marketplace. The more content there is, the more attractive the proposition becomes for both data contributors and to data consumers. In Streamr data market, a web portal (implemented as a Đapp) facilitates the discovery of what data exist out there, provides a comprehensive toolkit for the creation and management of data streams, and makes it easy to subscribe to data streams of choice.

2.4 Streamr Network

> Streamr Network is the data transport layer in the technology stack. It consists of Streamr Broker nodes which establish a P2P network. The network hosts a publish/subscribe mechanism and supports decentralized storage of events. The network throughput scales linearly with the number of participating nodes, and it can process millions of events per second.

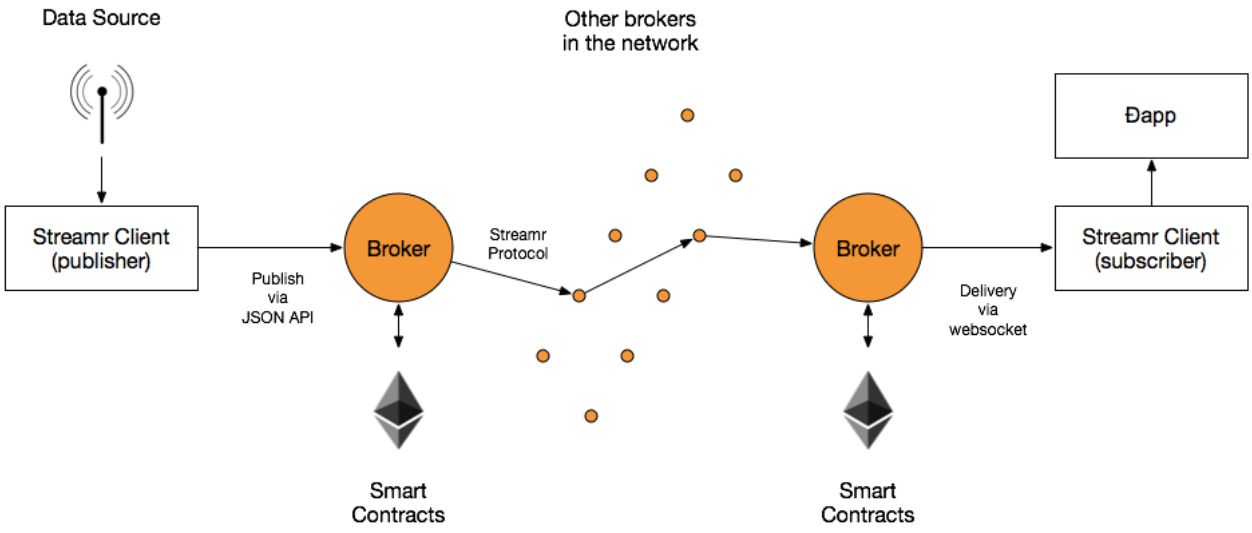


Figure 4. An example of an event traveling through the broker network from a data source to a subscriber Đapp.

The Streamr Network (Figure 4) is a transport layer of the Streamr stack. The network handles all messaging in the decentralized data pipeline. The layer consists of primitives (events and streams) and broker nodes. The broker nodes operate on the primitives, and the collection of broker nodes constitutes the P2P network which handles decentralized storage and decentralized messaging.

The infrastructure layer uses the underlying Ethereum stack for its operations. Node coordination requires robust consensus, which is implemented through smart contracts. The raw event data itself usually doesn't go into the blockchain, which together with partitioning allows the Streamr Network to scale to millions of events per second and higher.

The Streamr Network combines the best parts of scalable cloud-based real-time data transports (e.g. Kafka, ZeroMQ, ActiveMQ) and what's available in the decentralised P2P/crypto community (Whisper[6], Bitmessage[7]). The cloud-based frameworks use efficient sharding and persistence schemes to reach high throughput, but only in a trusted local network environment. The peer-to-peer protocols showcase effective strategies for routing, peer discovery, NAT traversal, location obfuscation, and so on, but fail to deliver the throughput needed for data intensive real-world applications.

2.4.1 Events

An event[8] is a timestamped piece of information. Every event contains headers and content. The headers specify the metadata for the event, such as its timestamp, origin, and content type. The event protocol supports arbitrary types and formats of content payload, e.g. JSON messages or binary images. The content type indicates what format the content is in. Event headers and content are encoded in a binary format for transmission.

All events on the Streamr network are cryptographically signed. All events have an origin, for example an Ethereum address. A signature is calculated from a private key and the rest of the message. The signature is used to prove the message origin as well as the message integrity. Since the event format allows for any kind of origins and signatures, the system is future-proof.

The following table lists the information contained in an event.

Field	Description
version	Version of the event protocol
stream	Stream id (Ethereum address of the stream smart contract)
partition	Stream partition (see section on partitioning)
timestamp	Event timestamp (ISO 8601)
contentType	Instruction on how to parse the body (e.g. JSON)
encryptionType	Encryption algorithm used to encrypt the content

[6] Gav Wood: "Whisper PoC 2 Protocol Spec" (https://github.com/ethereum/wiki/wiki/Whisper-PoC-2-Protocol-Spec)
[7] See https://bitmessage.org/wiki/Main_Page.
[8] Streamr events should not be confused with events in Ethereum smart contracts.

content	Data payload
originType	Instruction how to interpret the origin
origin	Data originator
signatureType	Instruction on how to interpret the signature
signature	Cryptographic signature proves origin and integrity of message

2.4.2 Streams

All events belong to a stream. There can be any number of streams, each of which groups together events that are logically related and stored in an ascending chronological order. Stream metadata is stored in an Ethereum smart contract. Each stream is identified by the Ethereum address of the contract. For scalability, events (i.e. the actual data points) are not stored in smart contracts or in the blockchain.

A data stream holds a set of permissions. The permissions control who can read events from the stream (subscribe), and who can write new events to the stream (publish). The stream owner controls the permissions, but she can also grant or delegate the permission control to third parties where needed.

The following table lists the metadata for a stream.

Field	Description
id	Stream id (Ethereum address)
name	Stream name
description	Stream description
owner	Stream owner
permissions	A mapping from Ethereum address to permission levels

2.4.3 Publish/Subscribe

Data delivery in the network follows the publish/subscribe paradigm[9]. Events published to a stream are promptly delivered to all authorized and connected subscribers of the stream.

[9] Wikipedia: Publish/subscribe pattern (https://en.wikipedia.org/wiki/Publish-subscribe_pattern)

Subscribing to streams can be restricted to certain users only, or be free to the public. Similarly, the permission to publish content to a stream can be held by one, many, or everyone.

The publish/subscribe paradigm enables many messaging topologies used in real-world applications:

- One-to-many (for example, a news channel or stock ticker)
- Many-to-many (for example, a group chat or a multiplayer game)
- One-to-one (for example, a private chat or an analytics pipeline)
- Many-to-one (for example, a voting system)

Note that publishing an event need not imply that the event is delivered to any clients: It may be the case that there are no subscribers. Still, the event is persisted and delivered to a number of broker nodes for redundancy.

Technically, there are two types of subscribers. The majority of the data flow goes to subscribers connected to the network via a direct connection to a broker node (see Section 2.4.4. below). They can be, for example, web front-ends of Đapps, event processing chains running on Streamr Engine, or IoT devices controlled by data from the network.

Smart contracts are a special type of subscriber supported by the Streamr Network. Broker nodes in the network are incentivized to deliver events to subscribing smart contracts. In this scenario, of course, blockchain scalability limits apply. The mechanism allows the network to act as an oracle, meaning that data can be pushed to smart contracts without help from a 3rd party oracle. Since all data in the network is signed at the source, it can always be verified and trusted.

2.4.4 Broker node

The Streamr Broker node is the core software component in the network. A broker node handles tasks such as publishing events, subscribing to streams, handling storage, and communicating with Ethereum nodes via JSON RPC calls. The broker node exposes its functionality to connected applications via APIs.

The broker API can be used from apps using standard HTTP and Websocket libraries in any language. For ease of use, we'll provide reference implementations in a number of languages. The primary client library platform will be written in JavaScript. It can be used to deliver data to web-based Đapps running in the browser as well as to back-end applications running node.js. A Websocket API handles event delivery from data sources to the network and from the network to client Đapps. For stream management, a JSON API is used.

The Websocket streaming API takes care of the following tasks:

- Authenticate a session
- Publish events
- Subscribe to events in streams
- Deliver events to subscribed clients
- Query historical events in streams

The JSON API exposes the following functionality:

- Create a stream
- Configure a stream
- Delete a stream
- Get info about a stream
- Find stream(s) by search criteria
- Publish events (alternative to Websocket API)
- Query historical events in streams (alternative to Websocket API)

Most of the traffic between brokers consists of event messages, but there is also traffic related to routing and peer discovery. An important coordination task between brokers is partition assignment, in which a reliable consensus must be achieved. This mechanism is implemented as a smart contract which leverages the power of the Ethereum network (see Section 2.4.5 below).

2.4.5 Partitioning (sharding)

Event traffic in the whole network is divided into independent parts called partitions. Each broker node handles the traffic belonging to a set of partitions. This is how scalability is achieved: not all nodes handle all the traffic. This is similar to the partitioning scheme found in e.g. Apache Kafka.

The partition for a particular event is calculated by hashing the stream id. This is a fast operation and done locally. Using the stream id as the partition key means that all events in a particular stream always go to the same partition. This allows the network to maintain the ordering of events within a stream, and to store them efficiently.

It may happen that a stream receives such a volume of messages that a single broker cannot handle them. In this case, an another round of partitioning is applied to the streams themselves, and the traffic within a stream is split to independent parts. In this case, we hash the (stream id, stream partition) tuple to assign the network partition, and the publisher provides the partition key which assigns the event to a partition within the stream. The order of events for a stream partition key is preserved.

The number of partitions in the network remains constant until automatically incremented over time. As described in the next section, there is a coordinator smart contract which controls

network partitioning. The number of partitions is proportional to the number of broker nodes participating in the network.

2.4.6 Node coordination

In distributed data systems such as Apache Kafka and Apache Cassandra, node coordination is usually achieved by using a component like Apache Zookeeper. There is a centralized process for establishing consensus in processes like leader election. Alternatively, some systems use manual assignment of coordinator nodes which have special privileges in the network.

In a decentralized network, such centralized or privileged components cannot exist. Instead, the Streamr network uses the underlying Ethereum network to establish consensus for node coordination in the P2P network.

The key coordination task is the assignment of network partitions to broker nodes in the network, and the maintenance of changes in this information when nodes appear or disappear. Instead of a centralized component like Zookeeper, this task is implemented by a smart contract: the *network coordinator*. The network coordinator contract is deployed on the Ethereum blockchain. Broker nodes find out the current network state by watching and querying the smart contract. Upgrading the network is achieved simply by switching to a new network coordinator contract.

Rebalancing the partition assignments is one of the tasks of the network coordinator contract. Only useful changes are made, and if there are none, the function does nothing. When the network is unbalanced, calling the function awards DATAcoin to the caller. This incentive ensures that network rebalancing takes place when needed.

The nodes assigned to a partition receive all the data for that partition. Some or all of them calculate rolling checksums on the data, and report the checksums to the network coordinator smart contract at certain intervals. In a large public network, there are enough nodes for each partition to make it difficult for them to collude. Partition assignment by the network coordinator smart contract is also difficult to influence.

2.4.7 Incentivization

Subscribers are the consumers of data in the network. DATAcoin, the network's usage token, enables subscribers to subscribe to streams. Other parties gain DATAcoin by contributing to the network: the broker nodes (the "miners" of this network), and the data publishers.

Broker nodes are incentivized to do two things: report checksums for their assigned partitions to the network coordinator smart contract (see Section 2.4.6. above), and deliver data to any smart contract subscribers (see Section 2.4.3). Both operations cost some Ethereum gas, paid by the broker. This cost is covered by DATAcoin the brokers receive for making the network function.

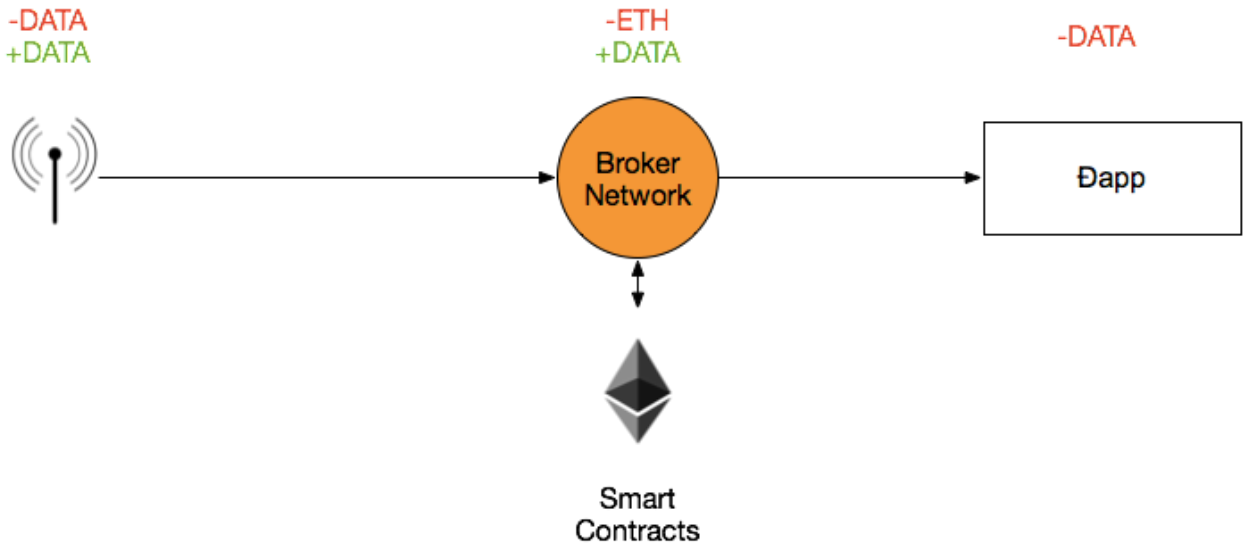


Figure 5. A schematic diagram of the incentive structure in the Streamr Network.

Checksums for a partition are calculated and reported by multiple broker nodes, and the brokers are rewarded only if the brokers agree on the checksums on a coherence threshold set in the coordinator smart contract (for example, 90% of assigned brokers need to report a particular checksum for it to be considered valid). If a node reports deviant checksums, none at all, or the checksums are not coherent, no reward is obtained and offending nodes become less likely to be assigned responsibility for a partition in the future.

As discussed, smart contracts can be subscribers to a stream. The subscriber sets a bounty in DATAcoin for delivering events to the smart contract. The bounty is collected by whoever delivers the data first. Usually this would be the broker node directly connected to the publisher, as that broker is in a forerunner position to make the delivery. Other nodes or external subscribers may watch this process and identify opportunities to deliver the data, if not delivered by the usual suspect.

A mechanism is also needed to prevent flooding in the network. A minimal cost must be associated with all publish operations as well as deliveries to subscribers. The network can aggregate the costs and commit every now and then to the underlying blockchain for scalability, similar to how state channels or micropayment channels work in some blockchain networks.

2.4.8 Event persistence

Events in data streams are persisted in the peer-to-peer network. This effectively turns the network into a decentralized time series database. The decentralization brings in a number of

advantages, including greater robustness, fault tolerance, lower cost, and the potential for anonymity.

With streams being sequences of events, the simplest form of storage is an event log. An event log can be stored in any block storage, such as a file system on the nodes themselves, or decentralised object storage such as Swarm[10], IFPS[11] or Storj[12].

For storage with much more granularity and querying features, decentralized databases such as BigchainDB[13] are emerging. A solution like this is a likely candidate for event storage in the Streamr Network. However the landscape is changing rapidly, and we won't commit to a specific storage solution at this time.

2.4.9 Data provenance

Security and data provenance are critical issues whenever external data is used as inputs to Đapps and smart contracts. As blockchain transactions are irrevocable, there's a clear incentive for honest parties to ensure that the inputs are trustworthy. There is also an incentive for dishonest parties — as well as unscrupulous hackers — to manipulate the data for monetary benefit.

In the Streamr network, every data point is cryptographically signed by a private key owned by the data origin. The area is undergoing rapid development, and many different methods are possible. Events could be signed with, for example, an Ethereum private key, X.509 certificate owned by an IoT sensor, trusted hardware enclaves using the Intel SGX chip and the Town Crier[14] relay, or a TLSNotary[15] service bridging data from a web API.

By design, the Streamr network is unopinionated on the method used to attest the data provenance, and it can indeed support any method available now or in the future. Events on the network always carry both the signature itself as well as information on which method to use to verify the signature. The client libraries which publish and subscribe to events can incrementally add support for different methods, abstracting away the inner workings of each method and making signature verification easy from a developer's point of view.

---

[10] Viktor Trón et al: "Introduction to Swarm" (http://swarm-guide.readthedocs.io/en/latest/introduction.html)
[11] See "IPFS - The Permanent Web" (https://github.com/ipfs/ipfs)
[12] See "Storj: A Peer-to-peer Cloud Storage Network" (https://storj.io/storj.pdf)
[13] See "BigchainDB: A Scalable Blockchain Database" (https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf)
[14] Fan Zhang et al.: "Town Crier: An Authenticated Data Feed for Smart Contracts", Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS), Vienna, Austria, October 24-28, 2016, pp. 270-282 (https://eprint.iacr.org/2016/168.pdf)
[15] "TLSnotary - a mechanism for independently audited https sessions", whitepaper, September 10, 2014 (https://tlsnotary.org/TLSNotary.pdf)

The initially supported signature algorithm is the same secp256k1 ECDSA as is used by Ethereum. This conveniently allows the network to reliably map any published data to an Ethereum address.

2.4.10 Data confidentiality

Given that anyone can participate in the Streamr network by running a broker node, event payloads of non-public streams in the Streamr network are strongly encrypted using asymmetric key cryptography. Only parties holding an authorized private key can read the data. The stream smart contracts hold the public keys of whoever is allowed to access the stream. At the time of publishing, the public keys of authorized recipients are used to encrypt the data so that only the authorized recipients can access the data.

Multicast encryption[16,17] techniques may be used to balance message size with keying complexity. Built-in encryption support also allows for straight-forward data monetization, as services such as the Streamr Data Market can be created to sell or rent access to stream contents. Publishers can rekey the data stream to selectively deny access e.g. in case they catch subscribers reselling their data outside the network.

A decentralized approach combined with encryption brings safety: The data is fragmented to a number of unknown physical locations, and it is protected by strong encryption while in transit and in storage. The design addresses the fears of companies and organisations that are concerned about the potential for compromised data via physical access to data centers and storage facilities.

2.5 Streamr Smart Contracts

A number of Ethereum smart contracts support the operation of the Streamr Network and the Data Market. The Streamr Network uses smart contracts for incentivization, coordination, permissioning, and integrity checking. The Data Market builds upon features provided by the Network for data licensing and monetization. DATAcoin, an ERC20 token, is used by both layers for incentivization, as a reputation metric, and as the means of payment.

[16] Micciancio, Daniele and Saurabh Panjwani. “Multicast Encryption: How to maintain secrecy in large, dynamic groups?” (http://cseweb.ucsd.edu/~spanjwan/multicast.html)
[17] Duan, Yitao and John Canny. Computer Science Division, UC Berkeley. “How to Construct Multicast Cryptosystems Provably Secure Against Adaptive Chosen Ciphertext Attack” (http://www.cs.berkeley.edu/~jfc/papers/06/CT-RSA06.pdf)

2.5.1 Stream

The **stream smart contract** holds information about a stream (see Section 2.4.2). Besides holding static information, it carries the permissions for the stream. In particular, it carries the public keys of authorized recipients for encrypted streams, possibly tied to a data license (see below).

2.5.2 Stream registry

The **stream registry contract** holds information about known streams in the network. Streams can be added to the registry for lookup purposes. The stream registry can also register streams on ENS (Ethereum name service).

2.5.3 Network coordinator

The **network coordinator contract** assigns partitions to broker nodes (see Section 2.4.6). Broker nodes register themselves with the coordinator and receive updates on the network state by watching the smart contract.

2.5.4 Data license

The **data license contract** represents a product listed in the Streamr Data Market. In return for DATAcoin, the contract grants access to an associated set of streams by registering the purchaser's public key with the streams. The data license can be valid for a certain period. After the license expires the buyer will no longer have access to new data published on the stream(s).

The raison d'être of a license contract is to hold the proof that the recipient has the right to access a data stream on specific and immutable terms of use, and to simultaneously guarantee that the data provider receives the agreed payment for real-time data as and when it is published. The terms of use may be stored in the data license contract either directly (and hashed as needed) or as a link to content-addressed storage such as IPFS. The contract may also contain proof about fulfilled legal requirements such as the result of a know-your-customer (KYC) process.

Broker nodes report event counts and rolling hashes to stream smart contracts, which in turn can report them to the associated license contract. The license smart contracts can implement almost arbitrary safeguards to prevent publisher fraud. They may, for example, lock the payment until a certain amount of events have been published on the stream. The payment could also be made incrementally over time or event-by-event as they are published. There may also be a mechanism for subscribers to flag bad data, negatively affecting the publisher's reputation (see Section 3 on DATAcoin and karma). These safety features ensure that the publisher cannot get paid without delivering quality data as promised.

3. DATAcoin

> DATAcoin is the means of compensation between data producers and consumers. It is also an incentive for running broker nodes in the P2P network. DATAcoin is the basis for karma, the reputation metric in the community. In a bigger picture, it is a way to gain exposure to data as a valuable commodity.

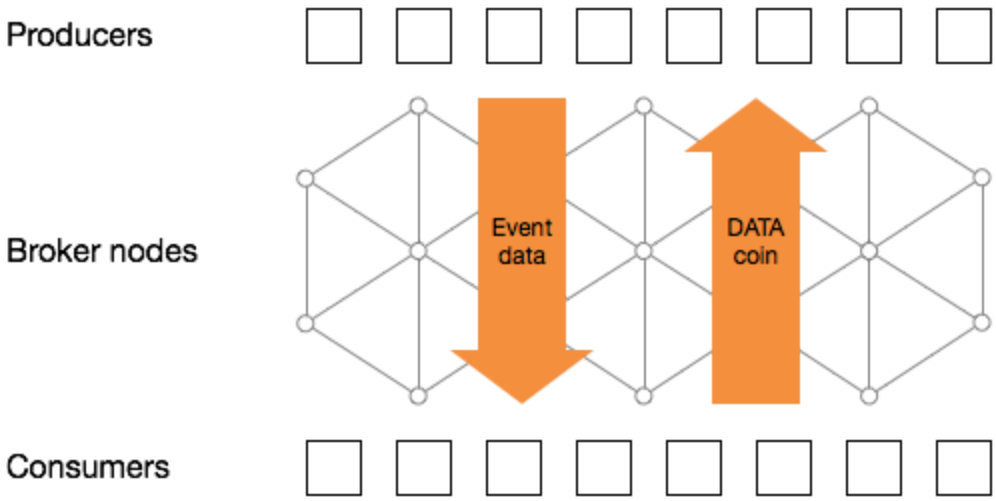


Figure 6. DATAcoin flows in the opposite direction from the data.

There's an integral role for a digital token in the decentralized data pipeline. DATAcoin is the usage token of the Streamr network. DATA is the symbol of the token.

- Maintaining and operating a P2P network takes resources: time, electricity, computing power, and communication bandwidth. Incentivization of the participating broker nodes are described in section 2.4.7.
- DATAcoin is the means of compensation between producers and consumers. In other words, it implements a monetization mechanism for data producers. This is an incentive for data vendors to step in and help grow the community to everyone's benefit.
- DATAcoin is the basis for karma, the reputation metric in the community of data producers, data consumers, and message brokers. Parties gain karma from DATAcoin transactions: Publishing data, consuming data, and running broker nodes that operate the network. A data producer gains karma when events she published are delivered to subscribers. Subscribers earn karma by receiving events. Broker nodes earn karma for helping with data delivery and persistence. Bookkeeping is easy: The amount of new

karma equals the amount of DATAcoin exchanged. The difference is that karma decays and eventually expires, while the token balance does not.

DATAcoin is implemented as an ERC20 token on Ethereum. The token smart contract maintains DATAcoin balances, and ensures that payments are handled in a trustless and secure way. Following the ERC20 standard ensures interoperability with wallets and other tokens.

DATAcoin will be created in a token generating event (TGE) currently scheduled for September 2017. Its details, terms and conditions, and detailed schedule will be announced later.

4. Current state

> There's a highly advanced platform already in place for creating data pipelines, visualisations, and off-chain computing microservices. The platform provides a functional starting point, but to reach full decentralization it must be refitted to run in a decentralized container and use the new Streamr Network layer for message transport.

We do not start from scratch. There's a functional and highly advanced platform in place for creating data pipelines, visualisations, off-chain processing, and Ethereum smart contracts. The software is built for the cloud environment with scalability, integrations, and fault tolerance in mind. Big data frameworks like Kafka and Cassandra are used for data handling and messaging. The Streamr platform was demonstrated live in EDCON in February 2017 and in various blockchain meetups since.

As to the pedigree, we created the first version of the software for our own use in algorithmic high-frequency trading soon 5 years ago. The principals all come from a financial background, being either quants, trading system developers, stat arb traders, and in some cases all of the above. Quantitative finance is a field where automatic processing of high volumes of real-time data has been the reality for the past 10-15 years and more. It is only the past few years where the same kind of modus operandi and the same kind of tools and platforms are finding their way into the world of IoT, IoE, and now into the blockchain space.

The current platform is functional, scalable, and in live use by corporate customers. Most of the components do not, however, translate directly to the new world. Storage needs to be decentralized, messaging, pub/sub functionality, and data monetization and encryption built into the transport layer, and the peer-to-peer network established along with node coordination and incentivization. The roadmap of how to do these things is presented in the next section.

5. Roadmap

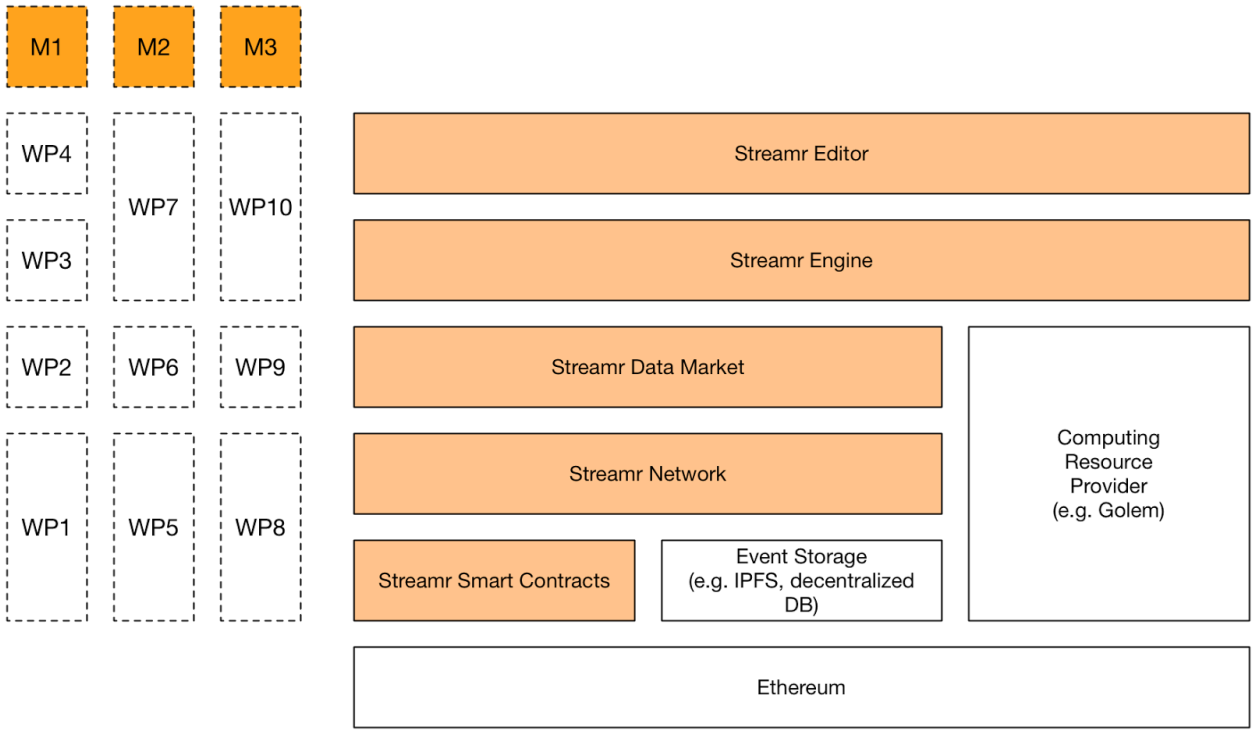


Figure 7. R&D roadmap for the Streamr project.

The roadmap (Figure 7) is divided in three milestones (M1-M3). Each milestone iteratively brings new features into layers of the stack. Each of the three milestones consists of work packages (WP1-WP10). Completing all WPs in a milestone completes the milestone. Each work package has a specific focus on certain layers in the stack. All WPs in a milestone will be worked on roughly simultaneously, but WPs in the next milestone will not begin before the current milestone is complete.

There will also be a security audit at the end of each milestone. All smart contracts will be audited, as will relevant parts of any non-smart contract code, for example the broker client itself.

We have chosen the iterative full-stack approach on the following reasoning:

- We can provide something that's working and usable to the community from day one.
- We are starting with an existing technology stack. The stack is modular so that any layers can be upgraded at any point along the road.

- Tangible and useful deliverables will be achieved regardless of which milestones we reach. The community won't be left high and dry with a half-way solution; there will be functional technology that fulfils many business use cases.

5.1 Milestone 1

Milestone 1 releases the first version of the incentivised data delivery network and the underlying smart contracts. Work on other layers builds on what we already have towards full integration with the Ethereum ecosystem.

WP1 (Network, Smart Contracts): Prototype decentralized broker network

- Create first prototype version of the decentralized broker node
- Integrate libp2p
- secp256k1 ECDSA signing and verification
- Event protocol
- JSON and Websocket APIs
- Stream smart contract
- Stream registry smart contract
- Network controller smart contract
- Partition assignment
- Checksum reporting
- Flood prevention using publisher/subscriber fees
- Basic DATAcoin reward scheme

The old "cloud" messaging layer and the new decentralized network will coexist side by side for a while, until the decentralized broker reaches scale and stability. The existing production Engine and Editor will run with the old broker until the network layer is upgraded.

WP2 (Marketplace): Barebones marketplace

- Discovery of publicly available data streams, initial categorisation and search
- Stream product implementation, buyable items which grant access to a set of streams in exchange for DATAcoin
- Users can define stream products and offer them on the marketplace
- Populate the marketplace with real-time data streams from different verticals such as the following:
    - Financial market data: stock prices, option prices, commodity prices, cryptocurrency rates, fiat exchange rates, trading volume, and corporate actions.
    - Social media data: Twitter, Instagram, Facebook, reddit, flickr, etc.
    - Transportation data: Departures, arrivals, geolocation, speed, heading for airplanes, ships, trains, and local transport.

    - Weather data: Temperature, precipitation, humidity, cloud cover, wind speed, both current and forecast.
- The first version of the marketplace will have some centralized “training wheels” to simplify access management and mirror the centralized/decentralized of the production network

WP3 (Engine): Engine goes Ethereum

- Streamr-Web3 bridge to support Ethereum interactions from Streamr
- Smart contract deployment, ABI support for predeployed contracts
- Local and transactional function calls
- Event watching
- Key and account management
- Support for different testnets and mainnet
- Signatures and signature verification
- Dockerizing the Engine for Golem or other decentralized computing provider

WP4 (Editor): Seamless on-chain and off-chain computing

- Improving the visual editor to fully support all Ethereum-related features in the Engine
- Integrated Solidity editor for writing custom smart contract modules
- A built-in selection of smart contract templates for the most common use cases (payments, betting, SLA monitoring, forecasting, etc.), and the ability to easily apply real-world data streams to these templates.
- Redoing the UI/UX of the editor and associated web application

5.2 Milestone 2

In this milestone, we launch the first version of the Data Marketplace, along with the features it needs in the underlying Network layer.

WP5 (Network, Smart Contracts): Support for data monetization and encryption

- First stable version
- Basic encryption support
- Data license smart contract
- Support smart contracts as a subscription target
- Add support for further data signing methods, e.g. based on X.509, SGX
- Basic storage in either decentralized block storage or decentralized DB
- Implement and utilize karma
- Stress testing, optimizing scalability

WP6 (Data Market): Completely decentralized Marketplace

- “Training wheels” removed from the initial marketplace implementation in M1 to achieve full decentralization, with data licenses modeled as smart contracts on the blockchain, and used for permissioning and access control
- Initial key distribution mechanism to support permission-controlled stream content on public network
- Seller identity verification
- Seller reputation mechanism

WP7 (Engine, Editor): Achieving decentralization

- Migrate to the new Network layer
- Deployment on Golem or other container provider
- Fault tolerance and failure recovery in decentralized environment

5.3 Milestone 3

The goal of milestone 3 is to reach the full Streamr vision. The R&D of milestone 3 is bound to change along the way, as the previous milestones will spark various ideas and the community will request new features. Milestone 3 will also contain significant marketing efforts to drive adoption of the stack.

WP8 (Network): Advanced routing, location obfuscation

- Location obfuscation
- Multicast encryption
- Stress testing, optimizing scalability
- Work on any problems found in large-scale deployments
- Work on features requested by community
- Work on integrations with emerging platforms

WP9 (Data Market): Community building

- Adding more data streams to the Data Market
- Stream wishlist, bounty program to accelerate adoption
- Improvements to reputation mechanism
- Increased community building efforts

WP10 (Engine, Editor): Productization

- Improving the UI and UX of the tools

- Onboarding, tutorial videos, help materials
- Add integrations to relevant platforms in blockchain, IoT, AI, or other spaces

6. Project management team

Henri Pihkala, M.Sc

Henri is a software engineer, a serial entrepreneur, and ex algorithmic trader. He has led the development of two high-frequency algorithmic trading platforms and designed and managed building the distributed Streamr cloud analytics platform. Henri is passionate about complex architecture, scalability, usability, and the blockchain.

Risto Karjalainen, Ph.D.

Risto is a data scientist and finance professional with Ph.D. from the Wharton School. He's a quant with an international career in automated, systematic trading and institutional asset management. Risto's interests include real-time computing, machine learning, evolutionary algorithms, behavioural finance, the blockchain, and fintech in general.

Nikke Nylund, B.Sc.

Nikke is a former low latency algorithmic trading strategist. He's got some 20 years of managerial and entrepreneurial experience as a founder and/or serial investor in ICT and tech companies with several successful exits under his belt. Nikke holds a BSc in Finance and Entrepreneurship from the Helsinki School of Economics.

Michael Malka, M.Sc.

Michael is an entrepreneur and technology enthusiast with 20 years of experience in different roles from software developer to CEO. He has been involved in software projects in different sectors from startups to banks and telcos. Michael studied computer science in the University of Helsinki before starting his first software company.

Advisors

TBA

7. Conclusion

This whitepaper outlines our vision for a robust, natively decentralized real-time data backbone for decentralized apps. We believe that the combination of a real-time data market and the data pipeline will be transformative for Ethereum smart contract developers and the Đapp ecosystem at large. Our ambition is to build a well thought-out and professionally implemented technology stack that serves the future needs of our audience, and provides unstoppable data for unstoppable apps.

Our technology stack is layered and modular, and it is built on a decentralized transport layer. There is a peer-to-peer network consisting of incentivized broker nodes. The network hosts a publish/subscribe mechanism and supports decentralized storage of encrypted events. Throughput scales linearly with the number of participating nodes, and the network can process millions of events per second.

The data backbone is an ideal facilitator of the M2M economy, where autonomous machines, bots, and robots buy or sell tiny pieces of data. The idea[18] has been raised that machines will barter resources such as storage, processing capacity, communication bandwidth, etc. We believe that using DATAcoin leads to much lower transaction costs than bartering.

Streamr is part of the computing revolution where monolithic solutions are being superseded by decentralized computing alternatives. In distributed computing, Golem replaces Azure Virtual Machines. In block storage, IPFS, Storj, and others replace Azure Blob Storage. In data pipeline and messaging, Streamr replaces centralized messaging and event processing platforms such as Azure EventHub and Azure Stream Analytics. There's a power transfer taking place from corporations and enterprises to individual citizens, autonomous agents, apps, and machines, leading to improved privacy, efficiency, resilience, and fault tolerance, and ultimately higher welfare for the good denizens of the connected society.

[18] Alex Puig: "How the Blockchain could enable the Machine Economy (M2M)", 11 January, 2016 (https://www.linkedin.com/pulse/how-blockchain-could-enable-machine-economy-m2m-alex-puig)