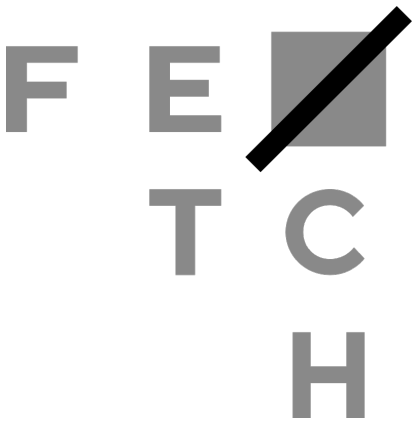


FETCH: TECHNICAL INTRODUCTION

A decentralised digital world for the future economy

Toby Simpson, Humayun Sheikh, Thomas Hain, Troels Rønnow, Jonathan Ward

Revision 1.0.7
Revised release, June 2018

https://fetch.ai
info@fetch.ai
https://t.me/fetch_ai

An Outlier Venture

Reader note: this document is subject to continuous change and refinement. Please ensure that you have the latest version before relying on information contained within it.

OVERVIEW

Today's digital world is increasingly sub-optimal and feels oddly, and somewhat ironically, disconnected given how "connected" we now are. It is full of under-utilised assets: hotel rooms lie empty, cars sit parked and unused for over 90% of their lives, half empty shipping containers travel the world, overcrowded, poorly optimised transport infrastructure eats at our sanity, power grids have peaks and troughs in usage that are shocking and organising even the simplest thing requires a herculean effort in hunting and gathering.

In short, we live in an increasingly complex world that we must somehow find ever more ridiculous ways to adapt to rather than one that figures out how to adapt to *us*, and, for that matter, itself.

Centralised systems are poorly placed to solve today's complex problems as they can't effectively work together to find solutions from the bottom-up. We propose an open, decentralised system where large numbers of simple things can cooperate—unsupervised—to solve problems without having had prior exposure to them.

Fetch achieves this by creating a decentralised digital world where a collective super intelligence actively delivers answers to you: a world where information that benefits you and information that *might* benefit you is delivered effortlessly. Fetch brings the world closer together and delivers power to the individual. It provides solutions to problems that no one has yet dared to address, enabled by a system that can handle huge numbers of tiny transactions conducted by our digital representatives.

We don't fix the old economy, we present an entirely new one.

TABLE OF CONTENTS

1. INTRODUCTION

Fetch is a decentralised digital representation of the world in which autonomous software agents perform useful economic work. This means that they can perform tasks, such as delivering data or providing services, and are rewarded with a digital currency for their efforts — the Fetch Token. These agents can be thought of as digital entities: life-forms that are able to make decisions *on their own behalf* as well as on behalf of their stakeholders (individuals, private enterprises and governments for example). Fetch's digital world is exposed to agents via its Open Economic Framework (OEF) and is underpinned by unique smart ledger technology to deliver high performance, low cost transactions. The ledger delivers *useful* proof-of-work that builds market intelligence and trust over time — growing the value of the network as it is used. Fetch can be neatly interfaced to existing systems with minimal effort, allowing it to take advantage of the old economy whilst building the new: plug existing data in to Fetch and watch markets spontaneously form from the bottom up.

Applications: The applications of such technology are many. By bringing data to life, Fetch solves one of the greatest problems in the data industry today: data can't sell itself. With Fetch, it can: data is able to actively take advantage of any opportunity to exploit itself in the marketplace in an environment that's constantly reorganising to make that task as easy as possible. Internet-of-things (IoT) devices inhabited by Fetch agents can increase utilisation by capitalising on short-lived opportunities to sell the information that they possess in existing as well as new markets: an agent in a vehicle can provide weather and road conditions by simply relaying the activity of its windscreen wiper and washer activity. Fetch's decentralised digital world enables the emergence of new marketplaces and allows this "unreal estate" to place relevant markets near each other for ease of exploration. The ability of agents to serve as representatives for data, hardware and services enables a better coordinated delivery of highly or even loosely connected services such as transport and insurance. Fetch creates a huge population of digital data analysts and sales agents who can work together, alone, or with human or corporate masters to reduce the cost of delivering complex solutions in our daily lives.

New opportunities: Fetch's autonomous agents actively push their value out to those who need it or who *unknowingly* need it. The Open Economic Framework provides a digital world for them to inhabit that grows in value as it is used: over time, the collective intelligence that is formed provides unparalleled guidance allowing for high speed, high reliability transactions. The network's expanding compute power provides all agents with the ability to gain new insights and understanding from their data. With machine learning technology integrated throughout the system, from the ledger to the agents themselves, it is a network that enables, encourages and deploys intelligence, and that actively creates new knowledge. Fetch provides the node structure, the OEF API, and agent Development Toolkits to make agents easy to deploy. Additionally, entire new industries can be built from AEA construction as opportunities exist to replace human intermediaries with trusted digital agents. Previously unprofitable data and services become valuable with Fetch, as the cost and friction of applying them is dramatically reduced. Data and hardware can now get up on their own two feet, get out there and sell themselves entirely free of human intervention.

Fetch Token: Fetch are issuing a fixed number of divisible tokens that are used on the network as the digital currency for all transactions, as well as for network operations such as secure communications. Tokens can also serve as a refundable deposit for both nodes and agents wishing to perform certain operations (as a security mechanism that discourages bad behaviour). As Fetch secures its foothold in the data, transport, services, and IoT industries its token grows in demand as each sector makes a larger and larger contribution to the Fetch network's economic throughput. An initial utility token will be issued on Ethereum for the token sale.

2. TECHNOLOGY IN THREE LAYERS

Fetch actively puts value-generating agents in contact with those that require it. The world is dynamically reorganised to remove friction from the process. Trust and reputation information is provided to allow users and agents to transact with the least risk. Our underlying ledger delivers a digital currency and a decentralised transaction system capable of scaling more effectively than alternatives, so that it can support many tens of thousands of transactions per second at virtually no cost. We enable improved utilisation of IoT devices and we bring data marketplaces to life. To deliver this, we have developed three key layers of technology:

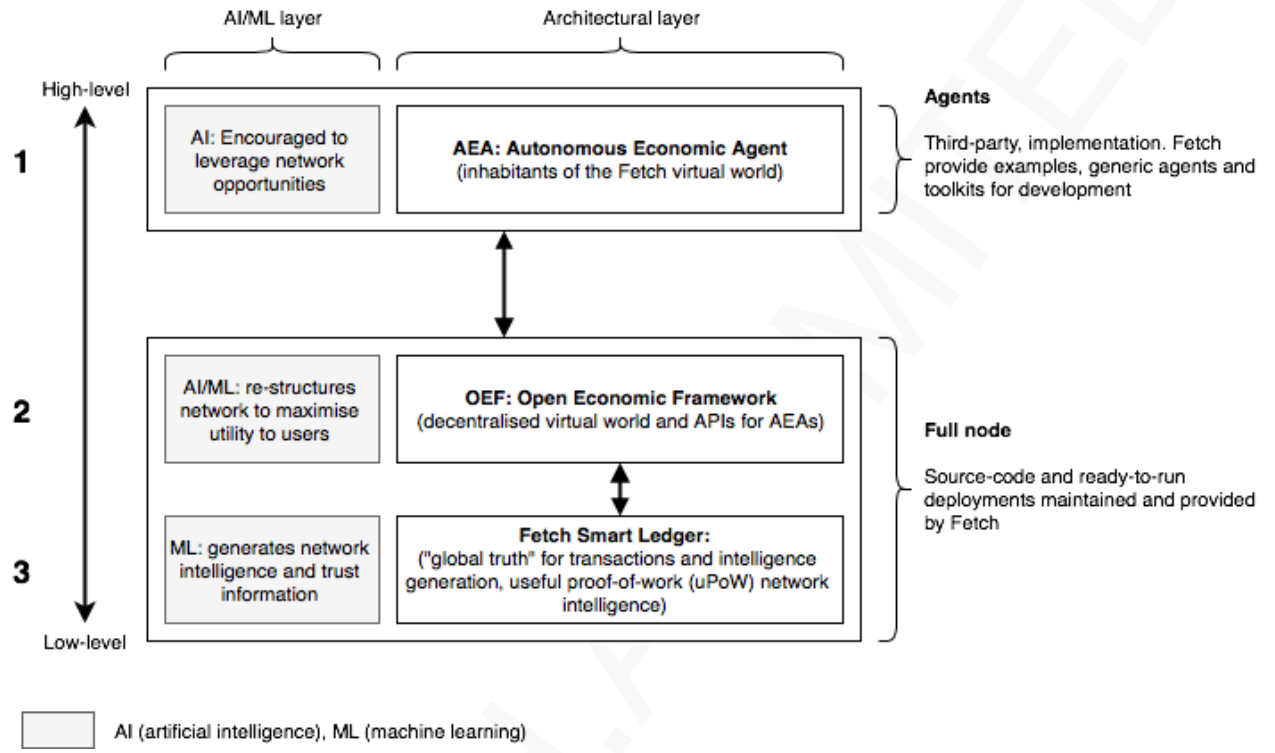


Figure A - Fetch's three layers. Layer 1 is the autonomous economic agents, AEAs, which live in the environment provided by layer 2, the OEF. Underpinning the OEF is the ledger that ensures the integrity of the global truth on the decentralised network and feeds the learning that provides trust, reputation and network intelligence. Layer 2 and 3 form a **node**. Fetch's peer-to-peer network is made up of many such nodes connected to each other in different ways.

Machine learning and intelligence is supported at all three levels. In the coming sections, we look at this intelligence stack and the AEA, OEF and Smart Ledger layers in more detail.

2.1 OEF: DECENTRALISED LIFE-SUPPORT FOR AUTONOMOUS AGENTS

The Open Economic Framework (OEF) provides life-support for autonomous software agents. These can be thought of as digital entities that are able to make their own decisions. They exist in a digital world that dynamically reorganises itself to present the optimal environment for agents to operate in. Primarily, this means rearranging space to place agents that can provide value "close" to agents that are searching for that value. Each agent therefore "sees" a different world from its perspective: a world that is reorganised for its perceived and declared needs. The OEF is the high-level node functionality: the layer on top of the raw protocol and ledger that delivers this environment and all the other operations agents need in order to go about their day-to-day work.

Figure B, below, shows the OEF's internal organisation and structure:

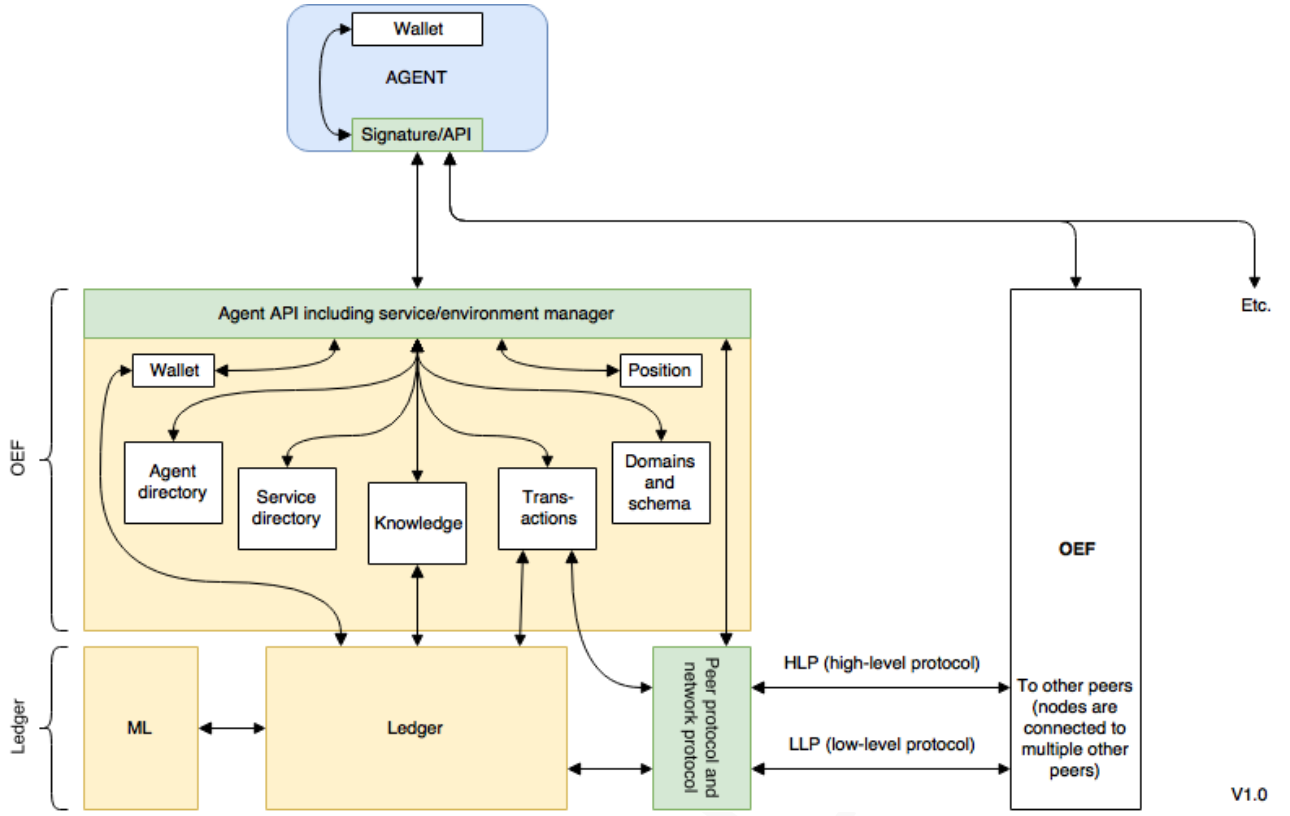


Figure B - Multiple OEF nodes collectively make up the decentralised environment. The two key APIs are shown - one for the AEAs to access their services and connect to/exist in the world and one for the peer-to-peer protocol which carries both high-level and low-level commands. High-level commands include operations such as agent transfer, exploration, search and discovery, whereas low-level commands consist of operations for consensus, peer trust, digital currency, transactions and protocol control.

The OEF's primary API, as exposed to agents, supports a number of base-level commands. Some of these commands are free, others require a small token cost. Other commands require what we term a "trolley token" — a small deposit in Fetch tokens that is refunded if the operation is gracefully completed. **Operation costs are decoupled from the Fetch token**[1] in a similar way to that of "gas" in the ethereum network. Thus there is a need to convert Fetch tokens to an operational fuel before commencing an operation. The node that performs the conversion receives the Fetch tokens in exchange for providing that service. The following table lists some of the commands supported by the OEF agent API:

Command	Cost	Deposit	Summary
Register	0	Yes	Register an AEA on a node
Unregister	0	No	Unregister an AEA from a node (refunds trolley token)
Register SN	1-10	Yes	Register for search notification (*)
Unregister from SN	0	No	Unregister from search notification
Ax node transfer	0	No	Agent move: move from one node to another
Communicate	1	No	Simple, secure agent to agent communication

[1] There is a constantly fluctuating conversion rate from the Fetch token to the Fetch operation fuel. Fetch operation fuel is used to pay the execution fee for operations on the Fetch network. This separation of high and low-velocity tokens allows large numbers of small transactions without the issues usually associated with such things.

Command	Cost	Deposit	Summary
Contract call	1-x	No	Perform a smart contract function call
Processing request	1-x	Yes	Perform trusted decentralised computing operation
Data store	0-x	No	Store data on external service (BigchainDB, etc.)
Transact	1	Yes	Perform a simple transaction (**)
Low-risk transact	5	Yes	Transact with simple escrow system
Trust enquiry	10-20	No	Perform a trust enquiry on node/AEA
Market-intel	1-x	No	Market intelligence enquiry
Advertise	0-5	No	Advertise available services
Search	0-20	Yes	Perform a network service search
Explore space	1-20	No	Perform an environment exploration (***)
Gather area	1-20	No	Perform an area gather for offline exploration
Opportunity search	0-x	Yes	Search for unfulfilled or badly categorised searches (****)
Construct ZKP	Unknown	Yes	Construct a zero-knowledge proof
Service discovery	0-10	No	Discover available services

*) Registers for notification about certain searches. This allows agents to be notified of and potentially respond to searches that fulfil the requirements on geographic and economic dimensions. The cost depends on the complexity of the filter. General "all" registrations are not generally permitted.

**) Transaction costs are *near* zero, but not actually zero: they are the lowest possible amount. When large numbers of transactions are occurring, this provides further incentive for nodes to provide high-quality service and to verify other nodes' actions.

***) Exploration is the "eyes" of the decentralised virtual world. With the appropriate filters and use, it allows agents to build a visualisation of what is around them.

****) Opportunity search allows the network to learn over time the correlation between requests and what was eventually used to satisfy those requests. It builds knowledge and facilitates real-time response to previous requests with previously unknown solutions.

It should be noted that the numbers presented in the above OEF command table are purely speculative, subject to economic modelling and are for illustration purposes only. A detailed technical yellow paper is being produced outlining the proposed on-ledger virtual machine and OEF commands.

The reward for providing these commands to AEAs is given to the node's operator. Figure B shows the wallet in a node. This reward encourages the node's operator to provide a consistent, reliable service and be positioned in an economically viable position.

Commands such as *Ax node transfer*, *explore*, *gather* and *search* allow agents to move around and view the environment in which they live. Trust and market enquiries permit agents to learn about the reputation and trustworthiness of other agents they encounter as well as the nodes that they are attached to. *Advertise*, *discovery* and *transact* allow agents to do business. The network supports simple and secure agent-to-agent communications and provides interface commands for large-scale data storage and recovery on third-party systems such as BigchainDB or other providers.

Agents need not restrict themselves to one node. Indeed, it is a wise move for an agent to have more than one footprint in the Fetch world. They can do this by registering on multiple nodes. This provides them with additional protection against failure, better trust information (as it comes from multiple sources) and higher performance as well as the ability to be in more than one place in the digital world at the same time.

Trolley tokens are managed in a smart contract on the system. These are refundable Fetch tokens that are required for some operations. The token is automatically refunded when the operation is complete or when the other party involved fails ungracefully. This is the "skin in the game" requirement for connecting to and using the OEF's network features.

2.2 ENABLING INTELLIGENCE, DEPLOYING INTELLIGENCE

Machine intelligence and learning exist in all three layers. The OEF and Ledger form the primary Fetch protocol: they operate on each and every node in the system and provide the environment in which the agents live. The agent layer is up to third parties to dictate and create[2]: we present a digital environment exposed via an API which provides opportunity for economic gain if utilised effectively. This effective utilisation, for scale and cost reasons, is best done digitally and without human intervention. This incentivises digital intelligence at the AEA level. At the protocol level, intelligence and learning are used to provide four layers of trust information:

1. Trust in how normal any given transaction is
2. Trust in the information received from other nodes on the network
3. Trust in the parties involved based on their history
4. Evolving market and data intelligence

These collectively provide reassurance to users about the trustworthiness of any given transaction. The ledger's useful proof-of-work[3] (uPoW) generates this information as nodes place transactions and transaction information onto it. This data grows in value over time as it is built from a larger and larger sample size. We can say with confidence that Fetch is a truly intelligent protocol: it uses this wealth of information to restructure itself dynamically in order to best suit its users as well as provide those users with information that allows them to manage their risks and transact at the most cost-effective rates possible.

Market intelligence is a particularly interesting aspect of Fetch. Over time, it learns more about what kinds of markets interact with others, under what conditions and which ones overlap with others. This data has previously been held in proprietary silos by very large online markets such as Amazon and eBay, but for the first time will be available publicly[4]. This hugely valuable information becomes accessible to participants in the marketplace as well as enabling smart market structure and an additional layer of information for agents to leverage in order to maximise opportunistic value use and increase utilisation of data and services.

[2] The Fetch foundation will invest to provide initial network intelligence and datasets to bootstrap the system, and will support third-party developers working in the Fetch ecosystem

[3] The network gains from all the proof-of-work activity. There is no "winner-takes-all" dynamic. Proof of work serves to place transactions and builds intelligence in the network for all its users

[4] Access to the data still requires tokens, and information about markets is not the same as the individual personal details of those involved in them, so user privacy is maintained

Fetch supports, rewards and encourages individual agents' intelligence whilst constructing a collective super-intelligence to support all users of the network.

2.3 GEOGRAPHY IN A DIGITAL WORLD: DOMAIN SPACES

Delivery of data and services in the real world is tightly coupled to location. Travel, meteorological data, insurance, local networking and information that is consumed is regularly related to physical proximity or in which direction it lies. Being able to sort, view, find, explore and consume such data and services according to its distance and location is a valuable concept — proximity improves relevance as well as access. In Fetch, our agents "live" in a decentralised digital virtual world. Whilst this world has all the advantages of being digital, it is organised in two ways: geographically and by economic purpose. Geographic organisation allows agents to be able to search in a particular locality. For opportunistic application of value, this greatly reduces the search scope necessary to find relevant agents. It also allows agents that inhabit hardware or represent human users to trivially filter by relevance and adjust according to, say, the mode of travel being used: the fact that a restaurant is 10 meters away but the user is on a motorway doing 120km/h with the nearest exit being 20km away makes it irrelevant, despite being nearby geographically. It is this smartness that is permitted by Fetch's digital world. Figure C shows how we take advantage of the fact that our virtual world does not exist in real space: we are able to stretch it, fold it and manipulate it to reel what you need in so that it is right next to you.

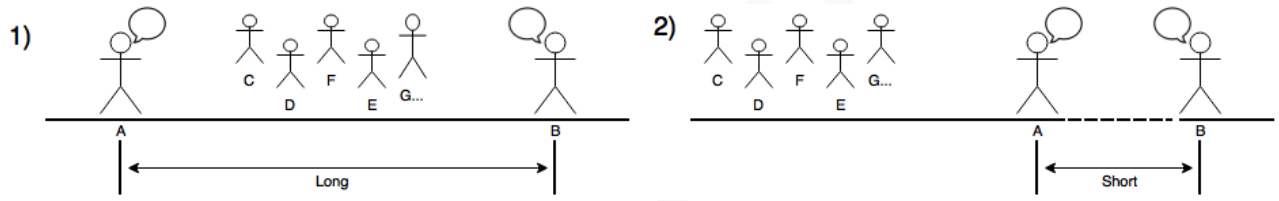


Figure C - Fetch presents an environment that reels what you need in so that it is next to you. We do this from both agent A and agent B's perspective, live, in real-time and for all users of the network. Agents C, D, E, etc. will all see what they need to. This effectively allows you to sit still and gather in what is useful and for the things that are useful to highlight themselves and position themselves closer.

In Fetch, agents exist at a position, but they *don't have to*. Their geographic location in the decentralised world is determined by the node or nodes that they are connected to. Should they wish to appear elsewhere, or indeed, simply ignore the fact that they have a position, then they are free to do so at no cost. No part of the network forces them to work only with the agents or nodes that are near to them. However, there is a significant sorting/filtering advantage from doing so. Regardless of an agent's spatial activities, relevant areas are still pulled in when necessary.

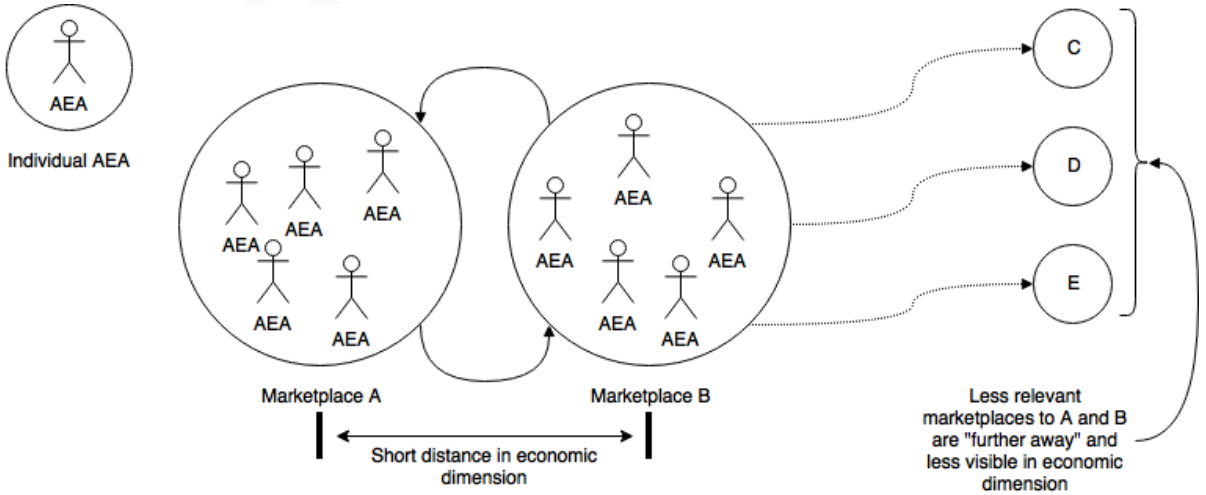


Figure D - Fetch Space has three dimensions; the geographic, the economic and network dimensions. These provide three entirely different ways for AEAs to position themselves, search and advertise. Many agents will exist in both geographic and economic space simultaneously to maximise opportunities available to them.

As can be seen in Figure D, there is also the concept of "domain space", or, the economic dimension. With domain space, agents that are economically close are grouped accordingly. This creates several dimensions to our virtual world;

1. **Geographical location** - agents can position themselves and then explore in a particular direction, across a range or via a radius/cone.
2. **Economic location** - relevant marketplaces and subject matters are positioned closer to each other than those that are radically different: Fetch introduces the concept of *marketplace proximity*.
3. **Network space** - the ability to explore the network in network space where geographical and economical location have no meaning.

A combination of these three provide enormous scope for smarter agents to best exploit the opportunities available to them.

2.4 ANATOMY OF AN AEA

The base level AEA (Autonomous Economic Agent) is a software entity that is able to perform actions without external stimulus. This makes it truly autonomous: it acts on its own behalf, not just on someone else's behalf. All AEAs must contain a unique identifier which comes from the agent's "wallet". This allows it to send and receive Fetch tokens. All AEAs must also maintain a list of nodes they are registered with and some basic statistics, all of which is signed and publicly verifiable. This can be seen in Figure E.

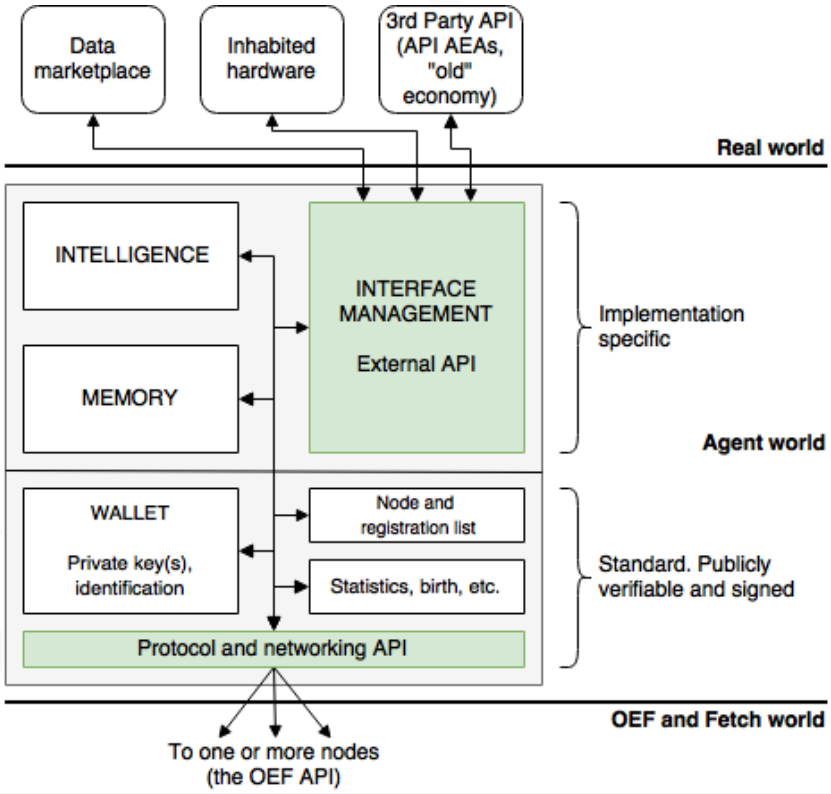


Figure E - Here we see the agent's relationship with the "other worlds", that of the OEF that provides its life-support and environment, and the other that is the real world. In this latter world, agents can connect to sensors, data, represent people or provide connections between the old economy and the new, e.g., ticket booking systems.

Agents are able to be connected to more than one node at once. They can do this for security and redundancy reasons and also to be in more than one place at one time (in any of the three dimensions:

geographical, economical or network). Node registration is required in order to participate. Registration requires what we refer to as a "trolley token": a token to be deposited in a smart contract that is refunded in one of two cases, 1) a graceful unregister with no pending/in-progress transactions or 2) the node fails.

Trolley tokens ensure that there is a token requirement in order to be part of the network and encourages good behaviour. It also attaches a cost to malicious agents: large-scale attacks cost tokens and all of these are likely to be lost.

2.5 AGENT TYPES

AEAs come in many flavours and the systems that support them deliberately encourage and reward improved digital intelligence. Whilst there is no such thing as a "typical" AEA, we envision at least five general application areas, although many AEAs will be made up of varying combinations of the types shown below as there is often significant overlap:

1. **Inhabitants** - these are AEAs paired to hardware that exist in the real world. These may be cars, drones, sensors, cameras, mobile phones and computers. AEAs do not control these devices, they exist "inside" them as controllers/operators/drivers: a digital version of what would otherwise be the human component. E.g. an AEA in a self-driving car does not drive the car, it tells the car *where* to drive.
2. **Interfaces** - these provide an interface between the old and the new economy. We call these API AEAs. They allow AEA entities to work with and leverage elements of the conventional economy such as ticket sales and can be thought of as *facilitation agents*.
3. **Pure software** - these are pure AEAs that exist in the digital space only. They explore, negotiate with, and find new ways of serving their stakeholders. These are teams of entities that work to organise, schedule and arrange other AEAs attached to hardware and interfaces to provide complete solutions.
4. **Digital data sales agents** - these are a specific class of pure software agents that attach to data sources in data marketplaces and go out into the Fetch world to extract value from that data. This is a solution to what is seen as the number one problem of the data industry: data does not sell itself. See 3.3 for a larger discussion on this application area.
5. **Representative** - AEAs that represent an individual and act as their interface to the Fetch network acting as a "digital butler". Their learning systems are involved with understanding preferences and tolerance for change whilst initiating autonomous requests to fulfil the requirements of the owner.

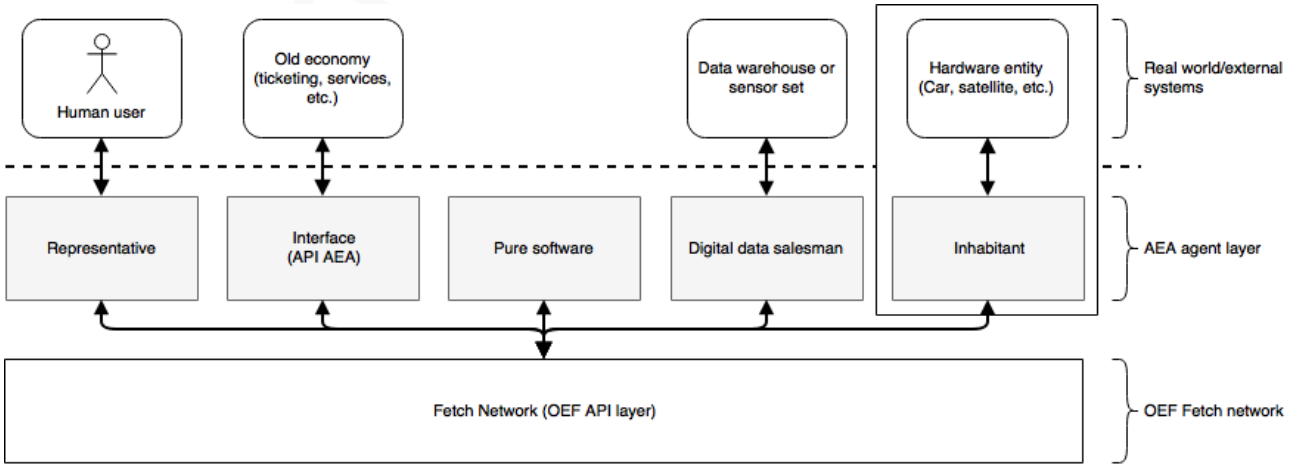


Figure F - Five different major classes of autonomous economic agents: those that work entirely internally to the Fetch network and those that interface to external data, hardware entities or represent human users.

2.6 DOING BUSINESS

Fetch is an effective, active brokering agency for putting agents who have value in contact with those that need it. It allows those agents to passively or actively be involved in the delivery or search but particularly rewards active participation though its multi-dimensional decentralised virtual world. The simplest conversation that two AEAs, connected to the same node, can have with the OEF in order to do business with each other is illustrated in Figure G, below. This shows the minimum message exchange between two agents in order to conduct a simple transaction. It does not show active exploration of the network or what happens when multiple agents are able to deliver the value to the agent that requires it.

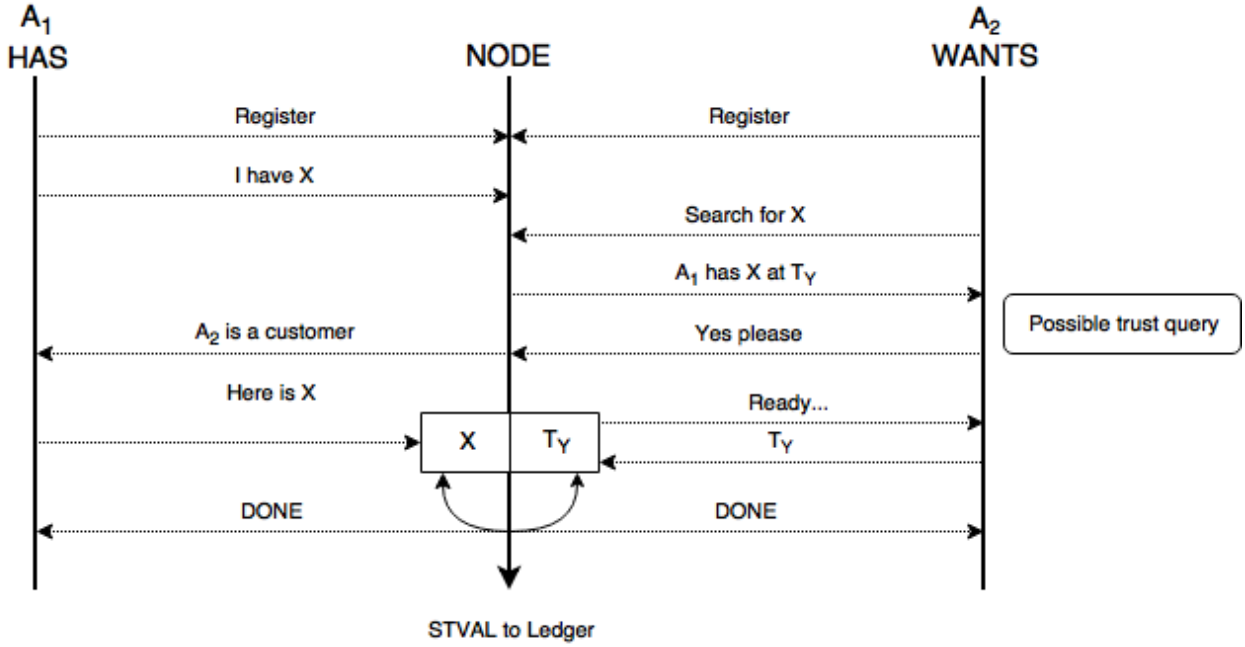


Figure G - In this example agent 1 is delivering data X to agent 2 who requires it. It is a simple search with a simple escrow transaction via the node to ensure that data item X and the payment (T) are both provided before the exchange occurs. The ledger gets five key pieces of information: source, target, value, action and location (STVAL).

The agent that is searching can refine the search by either the geographic or economic dimension to reduce the number of hops (and thus the cost) of conducting a relevant search. Figure H shows three different searches: a simple network search, a geographic search or an economic search. The risk with simple, unconstrained network searches is that the search hops will exceed the available tokens for the search before finding anything relevant. The agent can make better use of their exploration tokens by restricting the search with geographic and/or economic conditions rather than just the network's general organisation.

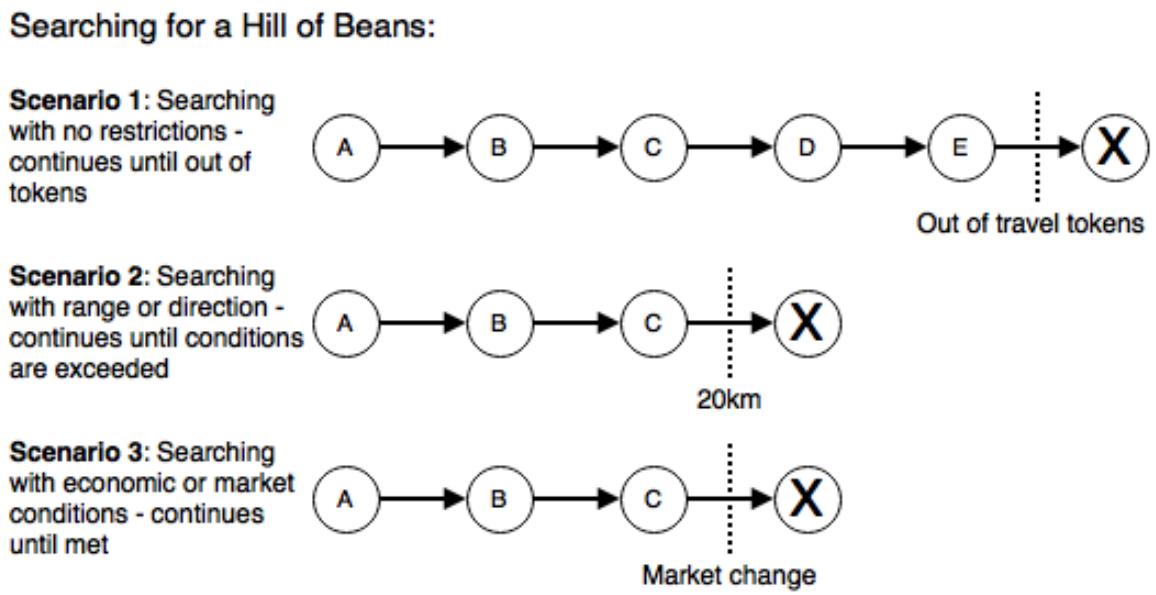


Figure H - A-E represent individual nodes. If an AEA is searching for a hill of beans, it helps to constrain the search by where this hill needs to be or by the beans (geographic or economic dimensions). This increases effectiveness and reduces the cost of the search and avoids endless explorations over irrelevant nodes.

3. THE FETCH SMART LEDGER

Fetch has specific requirements related to its unique architecture. Its ledger and underlying network needs to support the decentralised virtual world with a large volume of low-value transactions and the ability to compress the distant past without undermining integrity of data. It also needs to retain information about the transaction over and above that stored in traditional transaction systems due to the requirements of the useful proof-of-work (uPoW). Fetch's unique approach is therefore neither a blockchain nor a DAG, it is a unique blend of both data structures and incorporates many other innovations.

The technologies presented here are the working foundations upon which we are building. Fetch are working on a number of further innovations in this field which will enable a further generational leap in the ledger's capabilities, performance and stability. Furthermore, we anticipate new ways in which we can structure the underlying digital world and present knowledge and insight to the users of the network. As these new developments continue to be prototyped, trialled and developed, we will be updating our technical documentation accordingly.

3.1 THE LIMITATIONS OF EXISTING METHODOLOGIES

Traditional blockchain architectures lend themselves to intelligence-driven decentralised environments for agents because the nodes are largely static, remain on-line for extended periods of time and are capable of processing significant amounts of information.

A major limitation of blockchains is that blocks are added sequentially, so that only a single block can reference the previous block in the chain. This serial architecture is inefficient and greatly limits the rates at which transactions can be processed. Payment channel systems such as Lightning or Raiden[5] offer a way of providing scalability to blockchain solutions by taking large numbers of small payments off-chain. These systems can increase speed and decrease the cost of certain types of transactions but do not provide a general solution to the scalability issue.

In many unpermissioned ledger systems, proof-of-work serves as a means to control the block generation rate and to ensure significant computational power is associated with the generation of blocks, thus ensuring the fidelity of the network. The advantage of the conventional hash-based puzzle is that it is easy to verify and stochastic, so that a miner's success is proportional to his or her computational power. The disadvantage of this approach is that large amounts of energy are used in calculations that have no purpose other than securing the integrity of the blockchain. The difficulty of solving the hash-puzzles also encourages formation of mining pools, which decrease the system's decentralisation and therefore security. The Fetch ledger is designed to overcome these various limitations and meet all of the requirements of our digital economy.

3.2 LEDGER REQUIREMENTS

* **Scalability** — many millions of agents will be working alone or in groups to provide solutions for themselves and for other stakeholders. Without unconfined scalability, this will not be possible.
* **Stability** — for an economic system to be useful it is necessary to have a means of trading that ensures price stability. An important aspect of achieving stability is to separate fast moving tokens from slow moving ones. We believe that achieving this is crucial to the creation of a healthy marketplace.

[5] https://raiden.network/, https://lightning.network/

* **Useful economic work** — The original Bitcoin protocol uses proof-of-work to protect against consensus attacks such as double spends. This is a powerful idea, but we believe that rather than solving a puzzle with no other benefit, the computational power should be used solve relevant problems and thereby empower the economy.
* **Risk and trust information** — the network should provide trust, reliability, reputation and network intelligence information to allow users of the network to access the information they need to conduct business effectively and efficiently.
* **No loss of individual transaction data on the ledger's near past** — the machine learning systems Fetch uses require transaction information to be stored without rolling-up: each individual transaction's source, target, value, time, location and action from the near past is needed, but as time progresses, it is increasingly important to be able to compress proportionally to the age of the ledger's data[6].

Solving these issues is the key value proposition of the Fetch's ledger.

3.3 USEFUL PROOF-OF-WORK (UPOW)

Fetch's useful proof-of-work will involve the packaging of general-purpose computing problems into proof-of-work packages. These problems allow processing nodes with less computational power to occasionally earn block rewards. Verification of the subproblems will be carried out by nodes that "lost" the race for solving the problem with some smaller reward provided for these verification steps. Fetch will also incorporate tuneable PoW difficulties in relation to transaction fee so that nodes with low computational power can earn rewards by registering low-value transactions into the ledger. This distributed computing platform will be used to train machine learning (ML) algorithms, and will ensure integrity of the network by, for example, assessing trust in the validity of transactions and the ledger itself.

3.4 A SCALABLE LEDGER

Fetch's ledger is designed to allow for peer-reorganisation to present an environment that's efficiently organised for both spatial and non-spatial access and to provide large numbers of rapidly confirmed near-zero cost transactions. Its block structure contains transactions that include the action, timing and location allowing for useful PoW to build increasingly useful knowledge for the network to use to evaluate the risk associated with any transaction presented to the system.

The Fetch ledger is a novel structure that combines elements of a transaction chain with elements from a directed acyclic graph (DAG). Specifically we have used the DAG as an inspiration for parallel execution and transactions reference while requiring the strict ordering found in a transaction chain. That is to say, in the following we will describe our system where we are running N transaction chains (we refer to them as resource lanes) in parallel with a cross chain synchronisation mechanism. This mechanism is achieved by letting a single block appear in multiple chains.

We introduce the resource lane as a sharding scheme for ledgers. Unlike traditional sharding, in our system a transaction may be assigned to several different resource lanes simultaneously. We use a pre-evaluation module to determine which resource lanes are affected by the transaction. To this end, we group transactions by hashing the resource identifiers to minimise resource correlations. The systems capacity is directly proportional to the number of resource lanes available.

[6] This is additionally important because of the requirement to keep the requirements to operate as a node as low as possible: Fetch is inclusive and the large numbers of transactions will grow the ledger's storage requirements rapidly. It is essential that this growth does not "price out" nodes from taking part

We define a valid transaction as a transaction which, besides the typical transaction body, signatures contract references etc., additionally contains information about the resources it makes use of. That is, any attempt to access other resources than those described in the header of the transaction will be deemed invalid. In the figure below, we illustrate a system that initially has four transaction chains (horizontal lines). For the sake of simplicity we illustrate transactions (vertical lines) as entities that are using resources from exactly two resource lanes. However, a transaction may be linked to any number of resources as shown in Figure I.

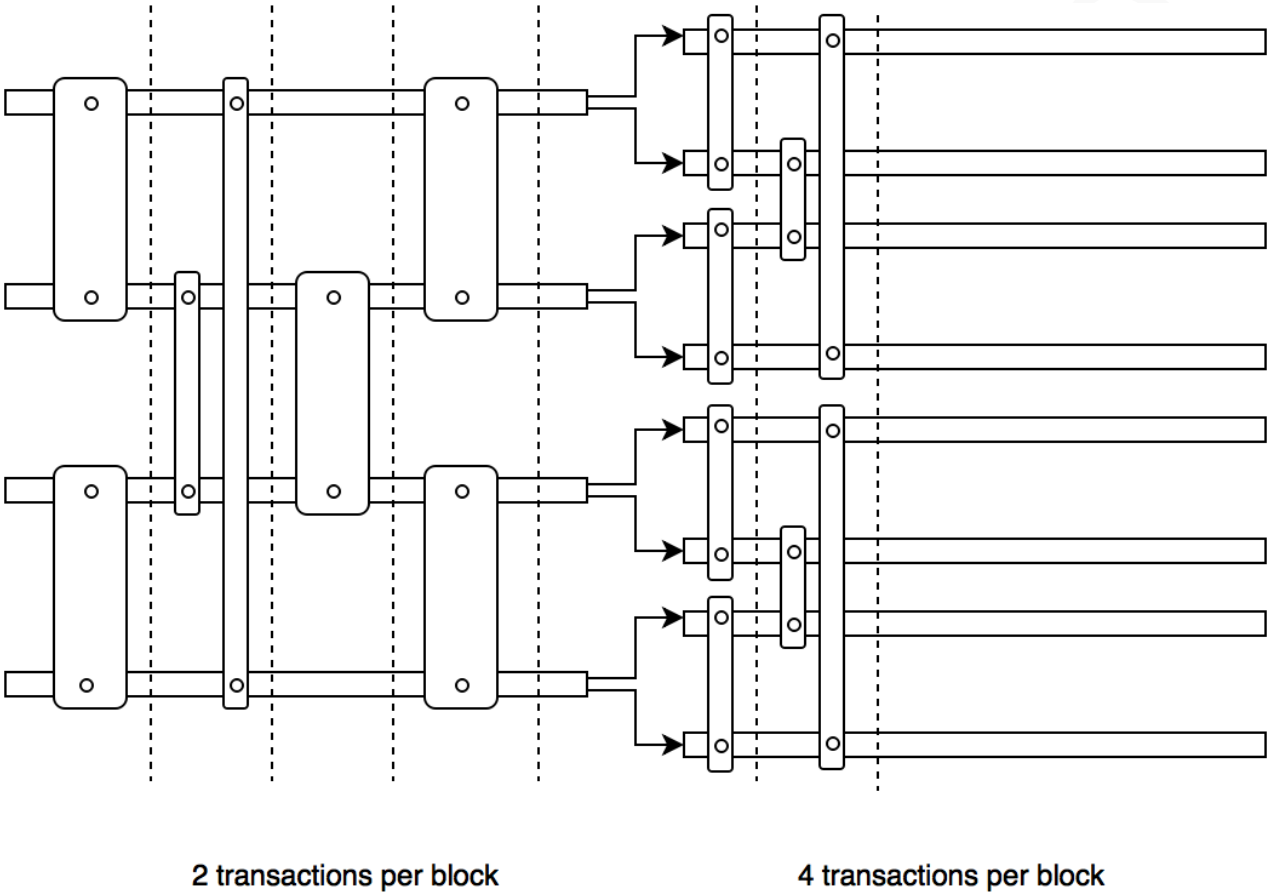


Figure I - Resource lanes for Fetch's internal arrangement of transactions to provide scalability. Horizontal lines represents resource lanes and vertical transactions. Circles illustrate transaction-resource lane relation.

As time progresses, the transaction capacity of the system can be expanded by introducing more lanes. This is done by the new lanes referring back to the old lane's blocks. It is very similar to chain forking except that we are doing it deliberately to increase the system capacity and with one difference: half of the resources are assigned to one fork and the other half to the other. At the time of splitting each old lane is referred to by two new lanes. The splitting process is achieved by inserting two new transaction blocks (each belong to separate chains in the expanded system) referring to the last block of the a single chain (in the unexpanded system). As an example consider the splitting of lane 9 into two new lanes 9a and 9b, as shown in Figure J.

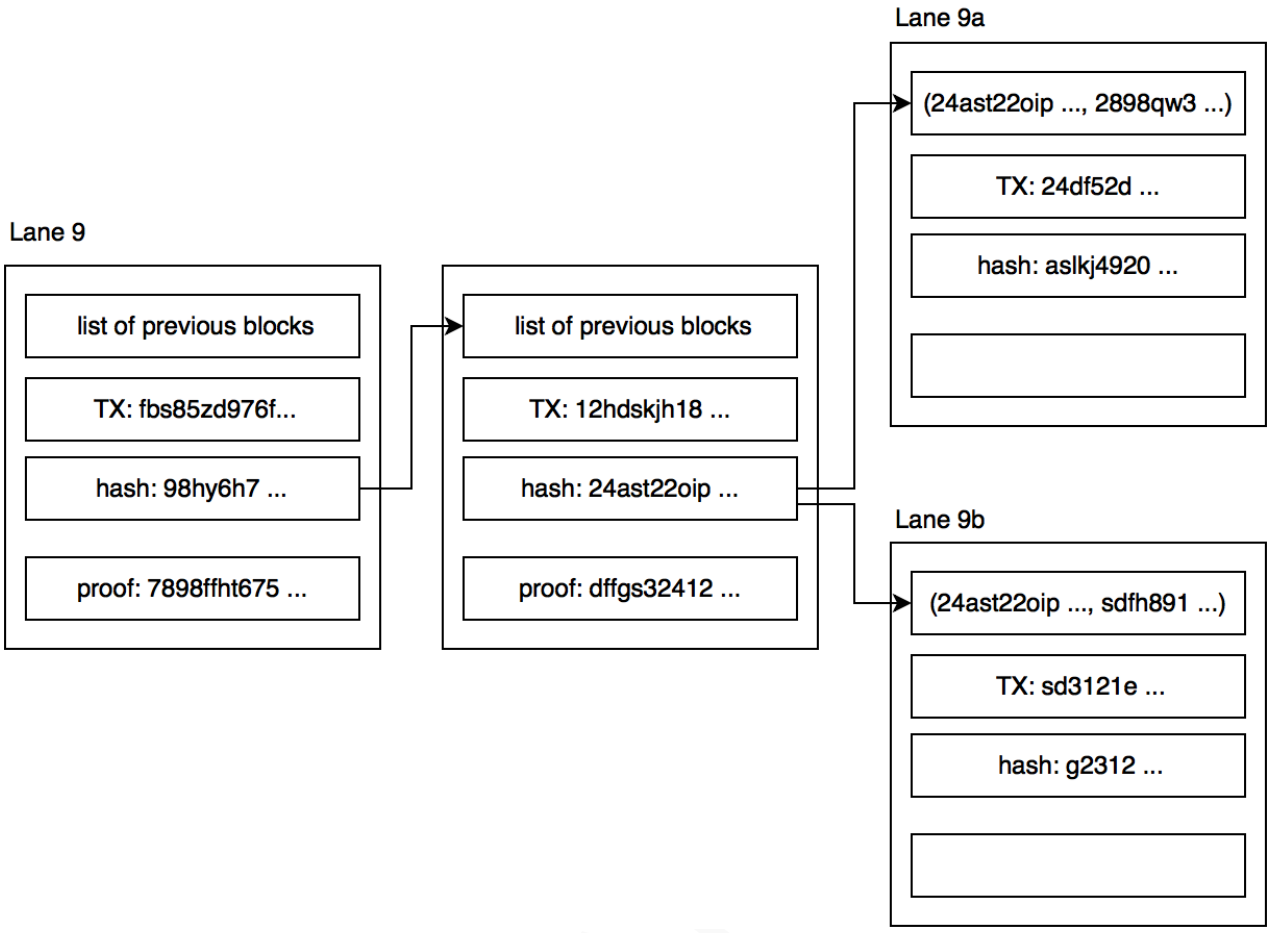


Figure J - Forking of resource lane 9 into two resource lanes 9a and 9b. This provides the mechanism for growing the capacity of the Fetch ledger.

In each of the two lanes, the first transaction's "list of previous transactions" both refer back to the last block of lane 9. In this way we track the origin of the new forks and by making use of a parameter based on the resource id, we can decide which resources belong to which chains.

As a consequence of this structure a single transaction will occur in up to as many resource lanes as the number of resources it is modifying on the ledger. Concretely, if the transaction "x7x8s922 …" is using resources from two resource lanes, its corresponding block will occur in both of these lanes. We illustrate this in Figure K:

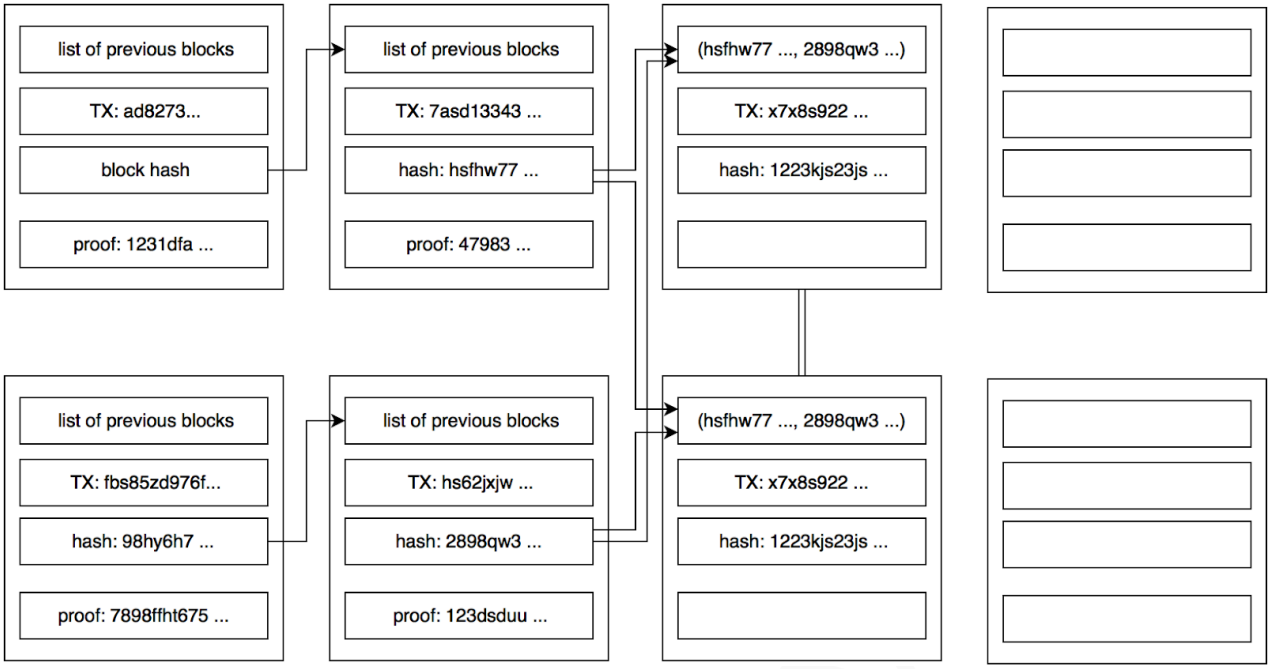


Figure K - Block appearance in multiple resource lanes. This provides a cross chain synchronisation method.

The block contains a list of previous blocks (from different resource lanes) and the transaction hash. We refer to this as the block's body. The block hash contains a single hash that is computed by applying a hash function F to the block's body. Finally, each block contains a proof similar to a normal blockchain. We will describe the consensus process in more detail in a later section. It should be noted that the two blocks are equivalent in the above figure and hence, only a single block is mined.

A block, in the current Fetch implementation, contains a single transaction and the system is in many ways similar to running multiple transaction chains in parallel with cross chain synchronisation. A valid transaction in our system consists of a list of resources, a transaction body, a transaction hash and transaction signature(s). Each resource is associated with a resource lane and we group the resources as described in the next paragraph, such that the first group of resources is managed by resource lane 1, the second group by resource lane 2 and the Nth group by the Nth resource lane. We depict this in Figure L.

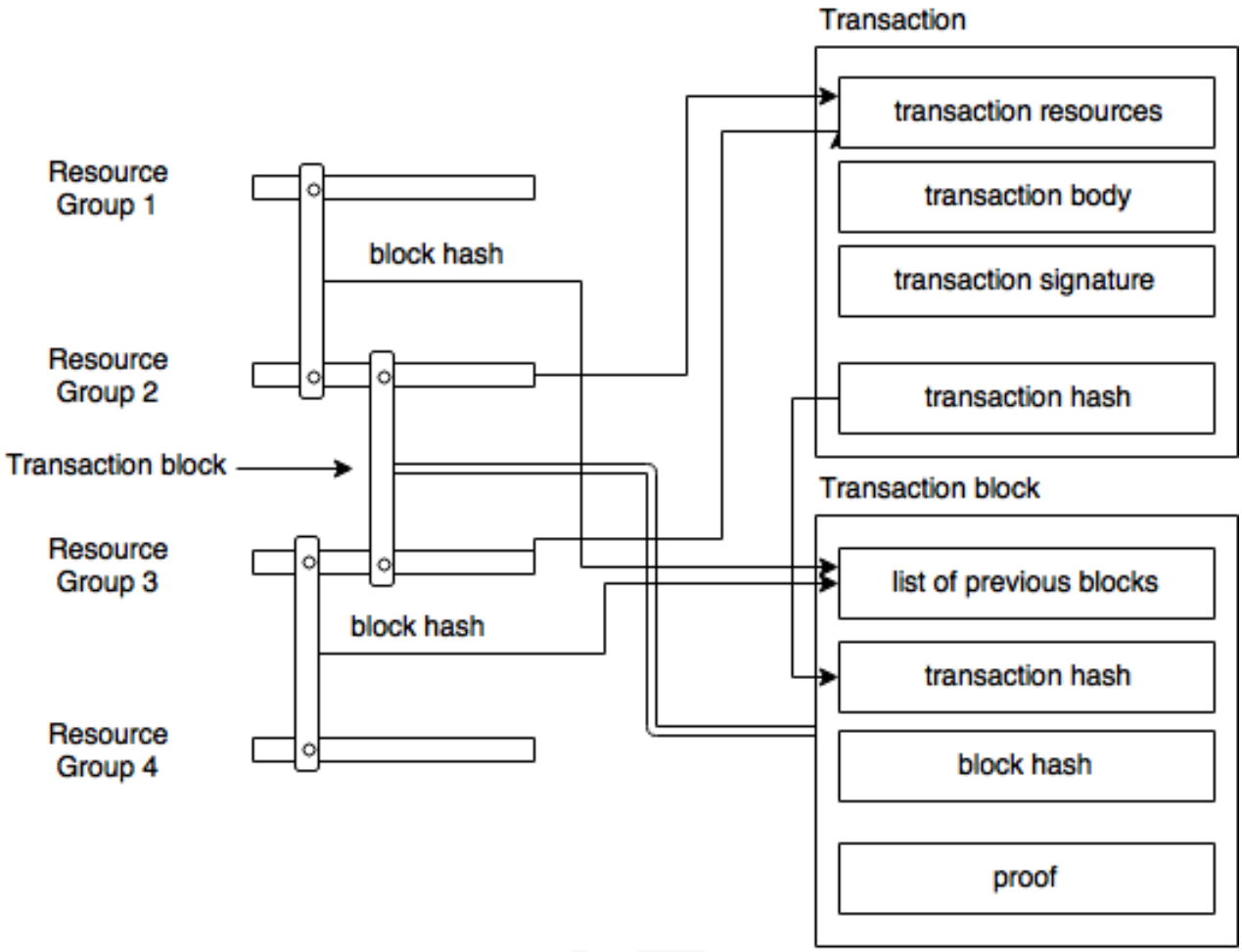


Figure L - Formation of transactions and blocks. Here we illustrate how the resource lanes are referred in the transactions and included into the block. Likewise all blocks refer to all previous, relevant blocks

We form a transaction block by referring to the previous block hashes in the groups that are defined in the transactions resources. By adding a reference to the transaction we can now compute a new block hash by simply computing the hash of the "list of previous blocks" with the transaction hash as H (list of previous blocks + transaction hash). Finally we add a proof for the block.

3.5 Intelligence about the markets for the markets

The Fetch network builds market intelligence, trust information and node reputation information in order to provide the users of the network with access to the information they need to maximise either their ability to find value or to provide it to those that seek it. This, combined with the three dimensions of spatial organisation (geographical, economic and network) means that data can be applied in ways that have never before been imagined. With the network delivering general-purpose computing, the data that is available can be transformed to deliver new insights and knowledge. Collectively, the network's inhabitants learn new correlations between requirements and how to deliver them. It is precisely this sort of emergent network intelligence that, for example, allows agents to establish climate and travel conditions from otherwise non-obvious sources such as the use of washers and wipers in all vehicles that are on the road.

3.6 Machine learning and intelligence

The Fetch ledger has two key requirements - to provide the agents that operate and interact within the Fetch network with support and guidance and to be secure. For this support, the Fetch ledger uses one piece of key information - the past actions undertaken by the agents. Each action is stored alongside

the value transacted and the public identities of the transacting parties. The action information can be used in novel and informative ways to support the trust in an agent's actions, to search for agents that can serve their needs, to supervise and understand their behaviour, and to dynamically form new domains and markets. To unlock these properties, it is essential to understand the public distributed ledger system as a stochastic process, where all transactions and actors are modelled with AI methods. Most notably, the actors are not only the agents using the Fetch network, but also the participants that form the ecosystem of nodes. By using a probabilistic framework, the nodes propagate belief about state and can find consensus by using their own version of the ledger history. The nodes can also establish belief in each other and, further, develop dynamic strategies that ensure fast confirmation cycles.

Machine learning targets three elements that enhance the performance, enable access, and provides trust: 1) understanding history: through ML models that capture behaviour, 2) understanding and planning the future: to understand how to distribute workload and increase convergence and 3) understanding the present: to distribute current belief and information.

The power that is unlocked from recording agent actions is considerable. Real world cost and even proof of work (or stake) strategies can be adapted to suit the transaction and participants at hand. The consensus-building itself allows the agents to use information about history and reputation, and the users of the system can choose to engage on the basis not only of simple contracts, but also of trust. There are several new elements that are worthy of discussion. Here we focus on several applications of machine learning to the Fetch ecosystem.

3.6.1 Belief propagation for ledger integrity

Fetch's novel ledger system enables large transaction volumes by enabling many transactions to be added to the ledger at the same time (in parallel). The main problem faced by this highly parallel system is double-spend transactions by malicious actors in the network. The detection of these fraudulent transactions consumes resources in checking the ledger, and it is mitigated in most systems by introducing a delay time before a transaction is accepted. The Fetch system will use machine learning to determine the probability of any transaction being replaced by an alternative double-spend, by propagating beliefs about the current state of the ledger between processing nodes. This time can be reduced further by using other properties of the transaction such as the sender and the number of past transactions between sender and receiver. The system will provide a disincentive for fraudulent actions by reducing the trust that the network associates with a particular economic actor.

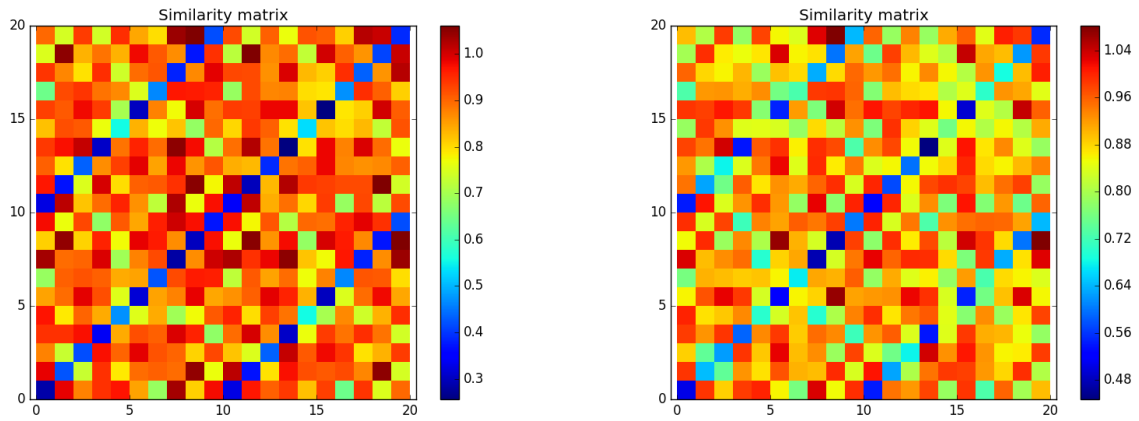
3.6.2 Interactions between the ledger and external economic agents

Since the ML algorithms that provide automated control of the Fetch ecosystem are likely to consume just a small fraction of the available computational power in the network, autonomous economic agents will also have the opportunity to submit problems of a certain form (typically a standard ML problem, such as image or speech recognition), and then pay processing nodes in the network for their effort. This mechanism also enables rewards to be tuned to the computational power of the network with a pricing mechanism used to prevent external processing tasks taking priority over the network's maintenance tasks. The probabilistic nature of transaction acceptance protocol also enables economic agents to, for example, immediately accept interactions with "trusted" economic agents and to accept the possibility of fraudulent behaviour if the transaction sums are small or are particularly time-critical.

3.6.3 Establishing Trust

The Fetch ledger automatically discovers the context of agents' actions and valuations and their relations with other agents. It does so by building up sequence models, i.e., models of temporal data that track agents' history and compress this information into fixed length representations that preserve the similarities that were present in the original data. By using Deep Learning methods, such as

recurrent neural networks (RNNs, LSTMs), this compact and low dimensional representation (also referred to as an embedding) allows an efficient comparison of different agents. With this information, one can establish a notion of an agent's reputation and of the consistency of a particular action with its previous actions, which will serve to predict its validity.



In both figures above, action sequences from 20 different users are used. Panel A displays cases for users with stronger occurrence of specific events than for those in Panel B. Embeddings of sequences were computed for each user. In both figures, embeddings belonging to users with similar action patterns are significantly closer in the embedded space.

Fixed size representations can also be used to map the free form description of the actions into a semantic space where similar expressions would be grouped together based on their meaning. For example, transactions related to different types of weather data would form a semantic cluster that could enable users to search for novel weather-related data sources.

3.6.4 Supporting Agents

A computational agent is considered a full learning machine that could embody anything from a human interacting with the system via a user interface to a fully autonomous reinforcement learning system. From the outset, the Fetch ledger was purpose-designed to support agents in all of their activities: finding each other, recording information, establishing a clear transaction decision record and enabling them to engage with each other. An agent can take multiple actions in a particular sequence and dynamically decide on the best strategy for success. Not only is this important for effective operation, interactions between agents also have an all-important negotiation phase where the traded action is established. Since agent interactions can be complex, the proposed action may manifest itself in an API-style program which is to be proposed by an agent. The Fetch ledger will store such information and provide means for checking the suitability of specific programs on the basis of historical use. This again increases trust and unlocks interaction of agents from different markets and domains.

3.6.5 Intelligent Interfaces

For economic agents to be truly autonomous, they need an understanding of how to talk to each other that goes far beyond merely knowing the appropriate communication protocols. They need to have an understanding of the purpose that a given message exchange serves. Given a description of a particular purpose, they need to be able to learn what message exchange they need to engage in in order to achieve this purpose.

Agents are able to draw on the records of previous interactions between agents to learn how the processes in question work. Any process logs summarising exchanges between agents will be stored alongside a natural-language description of the exchange's purpose. This could, for example, be an

instruction that was given to one of the agents by a human user. Agents may also be programmed to provide these natural-language descriptions themselves in order to facilitate human-machine interactions and to help agents develop their understanding of the purposes of agent exchange sequences. This enables agents to develop genuine autonomy in the pursuit of goals that can be defined with the same flexibility provided by a natural human language.

The result is a novel integration of approaches drawn from two fields that have not previously been known to cross paths: Natural Language Processing and Process Mining. Only recently have machine learning techniques been brought to this field, and where they have been tried, they have proven very successful.

A breakthrough in Natural Language Processing is known as *word embeddings*. As with other examples of embeddings described above, these are fixed-length vector representations of individual words. The process through which individual words are assigned to individual vectors is described as embedding the vocabulary in a linear (vector-)space. The embeddings uncover relationships between the words' meaning. If vec denotes the function that assigns words to vector, relations such as the following are observed:

$$\text{vec}('Paris') - \text{vec}('France') + \text{vec}('Italy') \approx \text{vec}('Rome')$$

We are taking these concepts of understanding into building intelligent interfaces for agents.

3.7 SECURITY AND ATTACK RESISTANCE

Fetch will be releasing a separate paper outlining details of the network's security and attack resistance. This paper includes results of Fetch's detailed modelling, our internal simulations and results of the first versions of the network under various attack scenarios. Fetch believe this to be a key topic and have designed the system with security in mind: from the useful proof of work through the trolley token mechanisms to other feedback mechanisms that incentivise good behaviour whilst discouraging (economically and otherwise) bad behaviour. It should *always* be more profitable to behave well on the network for any of its users. It should also be noted that Fetch have done detailed economic modelling of the ecosystem with its economic advisors from Cambridge and other institutions in order to provide further support for the network's performance and security.

The Fetch Security White Paper is expected to be published towards the end of Q2, 2018 alongside the OEF/uPoW Yellow Paper.

4. APPLICATIONS AND USE CASES

Fetch's autonomous economic agents are able to act without human intervention[7]. They can pay and be paid using the Fetch token. This has significant application: agents generally represent value of some form — either a data source, piece of hardware they inhabit or the person that they represent. Given the agents are able to search for opportunities themselves, they are constantly looking to maximise returns from utilising that value. In this section we talk about a few of the opportunities.

4.1 TRANSFORMING THE DATA INDUSTRY

Fetch aims to solve the greatest challenge for the data industry: that data does not sell itself and it is too expensive to use. Fetch allows data to be self-delivering and self-exploiting avoiding the key issues:

* **The cost of selling data exceeds what it is worth**. Much data has value, but that value is less than the cost of selling it.
* **Sales are time-consuming**. Data marketplaces are discovering that it is not enough to simply "build it and they will come".

Fetch drives more sales for datasets by allowing data to proactively sell itself: Fetch's AEA technology provides a mechanism for the development of low-cost digital agents that reduce the cost of sales to virtually zero. This unlocks previously inaccessible value and fundamentally changes data from being static warehouses of information waiting to be discovered into countless autonomous data containers delivering themselves opportunistically and directly to customers.

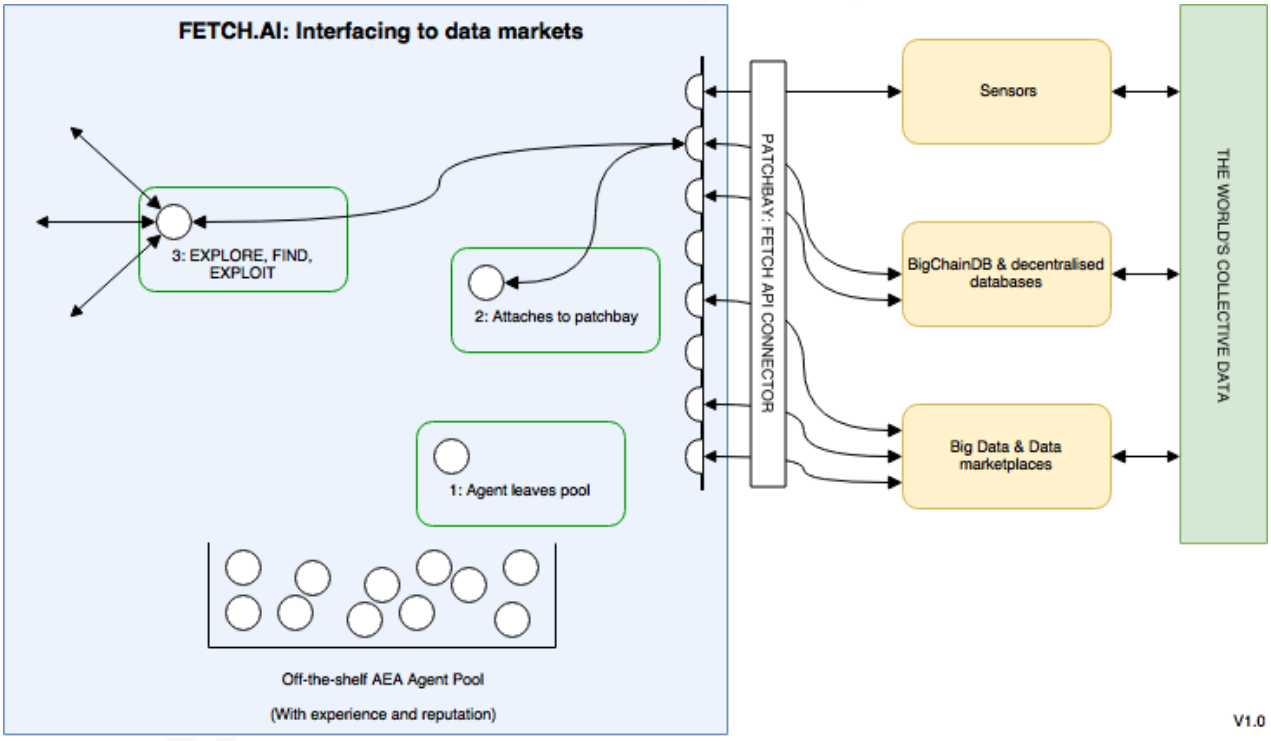


Figure M - A pool of standard AEAs, with pre-existing experience and reputation, sit in a pool ready to attach to individual data sources before exploring the decentralised world looking for potential users and application. They are able to scale up as demand rises and the "PatchBay" connector affords a simple, straightforward connection between data sources and the Fetch.AI network.

[7] Although they need not: AEAs can also act in real time under human control or supervision if appropriate

Large populations of cloned AEAs can be deployed en masse to respond to network requirements. The "Patch Bay"[8] technology interfaces multiple data sources to the Fetch network and makes that data available to all AEAs, not just the data provider's pool.

It should be observed that one of the key logistical advantages is also one that enables novel solutions: having all the data bought and sold on one system as opposed to it being spread across multiple services such as eBay, Amazon, Trip Advisor, etc., is significant.

Fetch will invest in datasets to provide an initial pool of useful information that can be utilised by AEAs. This includes supporting third party developers as well as continuing its work to form partnerships with commercial data providers and other information holders.

4.2 DECLUTTERING OUR LIVES WITH MORE OPPORTUNITIES

Data today is largely an exercise in pull rather than push. Unless it is known about, attached to and processed, it is largely static and cold. We see one effect of this on our mobile devices: a plethora of mobile apps whose job is often to merely process one data source into an application. These applications do not talk to each other, are limited to one application or application space and cannot combine the knowledge of other data sources in order to deliver more meaningful data to the user. Fetch allows a more useful combination and presentation of data by using populations of standard agents that can make data sources available on the OEF on demand. This information can then be presented in one application where much of the transformation and combination of data occur inside the Fetch network. Fetch's unique arrangement means that Apps that, for example, work on the basis of providing the user with local information, can combine context sensitive data without needing any prior knowledge of that data. This can be seen in Figure N.

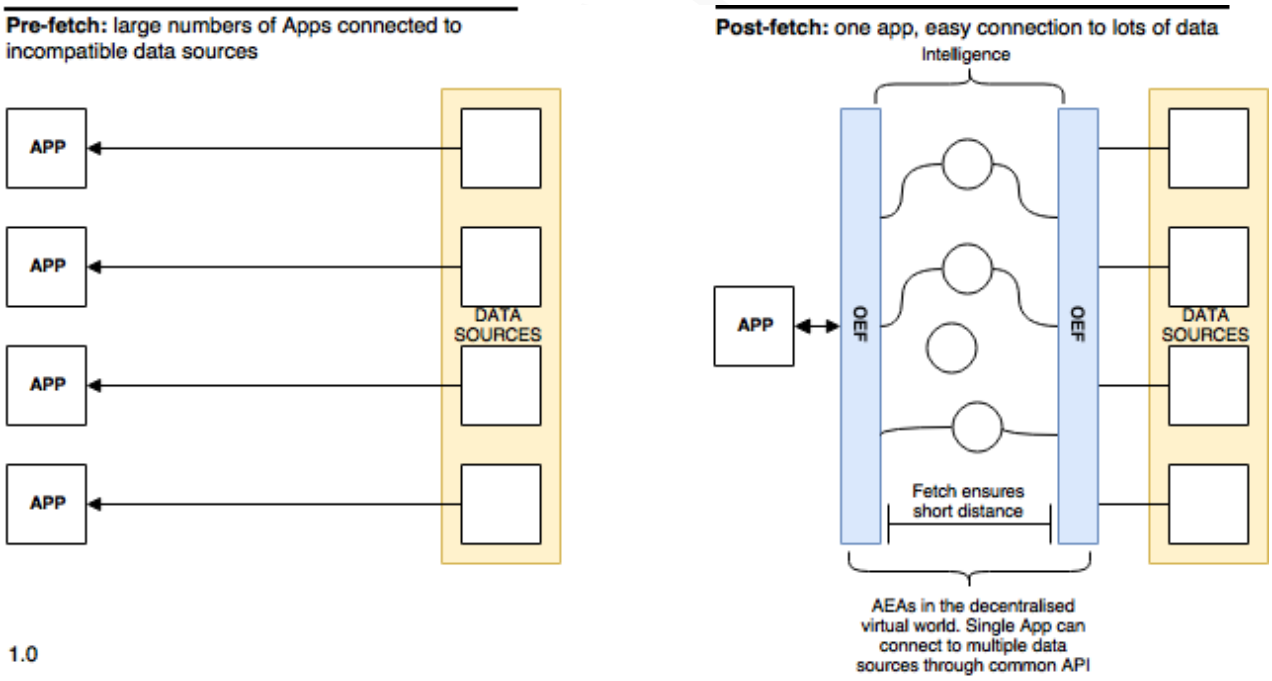


Figure N - A single App, with an AEA to interface to Fetch, can gather and present data from multiple sources without having to worry about many APIs and changing formats. Fetch dynamically organises the digital world to shorten the distance from source to target and turns the data gathering philosophy from "pull" to "push": the data delivers itself, lowering bandwidth use, waste and cost.

[8] Patch Bay, in this context, being a pure interface layer between Fetch and an external data source that allows agents to easily connect without having to know the details of the wiring behind the scenes

Imagine a service that presents parking information, charging points, wheelchair access, weather conditions, current restaurant table availability and review information of surrounding restaurants and does so entirely in keeping with what it had learnt about the user's preferences over time and was able to incorporate *new* information instantly without ever having seen it before. The Fetch powered universe allows static, previously unfindable data to be sold and utilised. Fetch does not replace data marketplaces, it interfaces to them and brings them to life.

4.3 A MORE JOINED UP APPROACH TO TRANSPORT AND ENERGY

Transport is one of the least seamless aspects of our daily lives. Each step of our daily journey is an exercise in pushing square pegs through round holes and yet we accepted inconvenience, largely without complaint. The individual components are largely centralised by different organisations that neither wish to nor benefit from cooperating with each other. Fetch takes control of this cooperation away from the individual companies and presents it on the network for use. This creates a considerably more joined up approach to transport than would otherwise be possible:

* **Availability of all relevant sensor information** - no prior knowledge is required of information sources or types for it to be potentially used.
* **Access to many transport sources** - API AEAs attach to existing ticketing and information services in order to present and make available those things into the Fetch network for all AEAs.
* **Information on the user's personal preferences** - personal "representative" AEAs know an enormous amount about an individual's personal preferences. This allows these preferences to be applied to all aspects of transport including dynamically responding to changes as they happen.
* **Agents actively working to deliver *useful* information** - AEAs are actively working to figure out how to deliver context sensitive, accurate and useful information to other agents. This is a positive intelligence arms race that encourages data transformation, deep analysis, novel uses of one form of data to generate another and more.

AEAs are collectively able to combine their intelligence to generate results that are considerably more than the sum of their parts: true emergent solutions to otherwise complex and hard-to-solve problems.

This approach has the ability to transform energy use and delivery. From shaping energy delivery to precisely match the requirements to incorporating personal or corporate preferences, Fetch has the power and the market data to make energy generation and delivery more efficient: a more fluid, less restrictive alternative to switching where suppliers and energy users benefit as energy grids can be managed with greater immediacy than is currently possible by dynamic price variation and other incentives in response to agent's needs.

4.4 OTHER APPLICATIONS AND DATA TRANSFORMATION

Agents are actively encouraged and rewarded by their ability to create data that is of value and deliver it to those that want it. Fetch reduces friction between the two that gives providers with vast opportunities to identify and exploit new markets. The gap between an opportunity being recognised and it being exploited is made very short: short enough to be jumped in near real-time. As has been previously mentioned, vehicle windscreen and washer data can be used to establish road network state and rain, but it can also be used in the summer to suggest the quantity of insects that are in the air. Such a correlation is learnt by the network's agents over time, but it is Fetch that allows the opportunities to be taken advantage of as they arise. Such an ability for an AEA to speculatively explore looking for possible answers creates a demand that can be satisfied immediately by other AEAs.

AEAs can also take advantage of the hardware they inhabit in other ways: constructing self-organising, self-healing, ad-hoc networks (from Bluetooth or spontaneous WiFi networks[9]) to allow for communications and messaging where none would otherwise be possible: enabling peer-to-peer wireless networking in areas that would otherwise not have network access and allowing agents to figure out how to make it work efficiently. By combining live location data, the concept of "I'd like to go for a run in as much privacy — but not out of sight of people — as possible, but with a restaurant at the end which is serving food and not too busy" becomes a viable delivery to a person by a representative AEA.

Agents can act on their own behalf to respond to the network's population's needs and we imagine that agents will be created to specifically solve problems such as intra-ledger transactions, currency conversions, access to financial, insurance and real-estate markets as well as generating new data that can be actively used rather than sitting waiting to be found.

4.5 REAL-TIME MARKET SHAPING AND CONSTRUCTION

As introduced in 2.2, Fetch constructs market information as a result of ongoing network use. This information exposes intelligence about market sizes and interactions that are held privately by companies such as eBay and Amazon. This includes: market sizes, interactions, overlaps, temporal and spatial data about such intersections and considerable information about transaction rates and sizes. This is data that is created as a fundamental part of the Smart Ledger's operation and is unique to Fetch.

Fetch allows its agent inhabitants to organise themselves to suit market requirements. It also ensures that collections of AEAs that form market areas can position themselves to be best placed to interact with relevant other markets. The AEA's position will change continuously and in real-time. Such a re-arrangement of markets has not been previously possible in the digital world: Fetch provides both the layered virtual space and the intelligence built from ledger proof-of-work to make this possible. Figure D in 2.3 shows economic space's organisation.

The market information and ability to respond to requests for value that is not currently in existence on the Fetch network provides agents with the opportunity to fulfil the request in near real-time. This knowledge of the correlation between request and what satisfied it is learnt on the network for the benefit of all future network users. This data knowledge combined with market knowledge is incredibly valuable as a product in its own right and to increase the effectiveness and margins for the AEAs in the system.

[9] The construction of spontaneous mesh networks is potentially a huge step forwards for a decentralised Internet

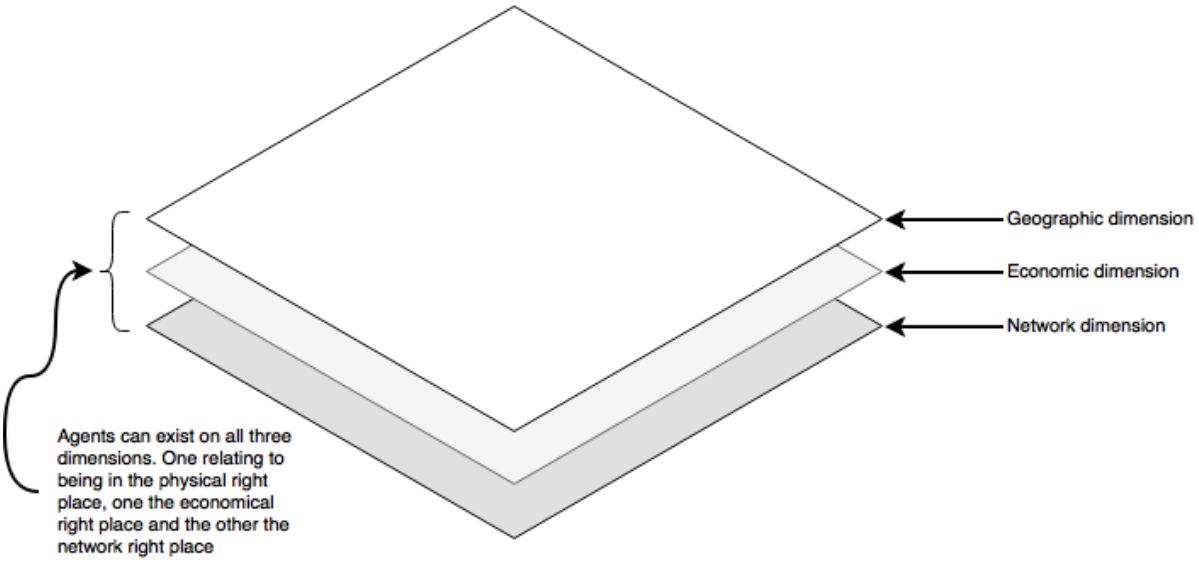


Figure O - In Fetch, an agent can be in the right place in three separate dimensions to maximise its ability to exploit its value or find someone to satisfy its requirements. It can move independently on all three layers and can even be in multiple places at once by connecting to more than one node.

4.6 Sample use case

Taking the example of a simple weather station IOT device, there is a sequence of events and activities required in order for it to work on the Fetch network. A simplified example of the minimum communication and actions involved in performing a Fetch transaction is presented in section 2.6, Doing Business. Here, we take a more detailed look at the steps involved including the development actions required to integrate a device that was not previously exposed to the Fetch network. At all stages there are many options depending on the preference of the device's owner or the type of device. This section does not exhaustively cover those options.

4.6.1 Creating an AEA

There are many ways of interfacing a weather station device to an AEA. These include:

* **Integration into the device itself** — this involves adding software to the device so that it can act as a standalone Fetch compatible unit. Fetch provides example source code that allows such code to be created but it does require the ability to run software on the device itself.
* **Interfacing a generic to a standard API/interface** — many such devices have interfaces to allow them to plug into a computer. This is most often a USB or Bluetooth LE connection. An AEA can run on a separate computer, either a small device such as a Raspberry Pi[10], a mobile device or a desktop computer and can collect this data and connect to Fetch.
* **Exposing device to data marketplace for generic AEAs to access** — the device itself or the computer it interfaces to can simply push this data to a data marketplace and allow AEAs to connect to that marketplace. In this case, the device is not an AEA in its own right.

The first two options create an AEA in its own right that is controlled by the owner of the weather station. The last simply relays to another service that is connected to Fetch and is of less interest for this example, but shows that "indirect" connections to Fetch are possible.

4.6.2 Transacting on Fetch

An AEA registers on Fetch on one or more nodes, each registration with a trolley token which is held in a smart contract that ensures that it is returned if the AEA disconnects gracefully or if the node fails

[10] The Fetch code is sufficiently light and resource efficient to run on tiny embedded devices, even down to the size of an Arduino with only a handful of tens of kilobytes of memory

or forcibly disconnects the AEA. Registration places the agent on the network and allows it to place itself in economic or geographic space if it wishes. The AEA can then advertise, actively explore the Fetch world for customers or be notified of potentially exploitable opportunities or a combination of all these.

1. **Advertise** — this is the simplest case and just pushes the market areas that the AEA is in to the network. In this case, it would be meteorological information that includes, for example, temperature, humidity, precipitation state and pressure. Other agents searching for such things would find the AEA if the geographical or market searches encompassed it.
2. **Actively explore** — the decentralised virtual world offers a way for active searching of space. This finds other AEAs that may wish to achieve something and an opportunity to push that value for consideration even if it would not have been triggered in a simple search as in option 1.
3. **Notifications of opportunities** — potential users of our weather information may not specifically search for such things, they may be planning a journey, a special event or something else. The AEA can register for notification of such searches as well as registering for notification of *failed* searches that are worth looking at.

Fetch is facilitating the connection of two agents and providing several ways of making best use of the decentralised virtual world.

5. DEVELOPMENT AND DEPLOYMENT

Fetch are developing the entire full-node software stack. This includes the OEF and the ledger systems along with all other software necessary to download and run a full Fetch node. The source code will be released as open-source and can be downloaded, compiled and run by anyone. Fetch also provide pre-compiled self-installing node packages for many platforms. The Fetch node software is written in C++ for performance and stability.

Additionally, Fetch are developing a number of AEAs to assist in bootstrapping the network and to provide a range of examples for developers. These agents include "stock generics" (those that can trivially attach to a data source and then exploit that data with a little user-configured context information) and a number of examples of the different agent classes we imagine (see 2.5, "agent types"). Very efficient "tiny-agents" will be developed for integration into internet-of-things devices. The AEA software developer kit provides framework applications for agents in several languages and is freely available to all agent developers. The SDK also contains examples of AI development (including integration with common ML/AI frameworks) and OEF environment exploration (on all three of the dimensions: economic, geographic and network).

Fetch provide a native wallet, AEA monitoring and exploration application for iOS, Android, Windows, Linux and MacOS. The source code for these applications will be available. These are first-class applications that are available from the relevant platform stores (where applicable) and require no additional frameworks, software or actions to be installed and used. If the users of these applications chooses, they are able to operate as a thin or full node. This provides them with income from providing OEF services, transaction processing and market intelligence. Furthermore, the wallet can configure itself as an AEA to deliver information to the network, or act as a relay for mesh networking (assuming the user gives permissions for these activities).

AEAs are able to interrogate the OEF for market and data intelligence. Example agents that perform these enquiries will be provided as they allow for access to information that leads to smarter, on-demand deployment of agents from agent pools to respond to specific spikes in demand.

5.1 DEVELOPMENT PHILOSOPHY

We develop high quality, robust, well-documented and efficient software that is built to last. The primary node software is written in C++ with minimal dependencies and is both memory and processor efficient as well as being designed to minimise network traffic to allow for maximum scalability with minimum resource use. We create a protocol with the knowledge that every byte saved —sent/received, processed or held in memory—affords the Fetch network greater scale and allows it to operate on more devices than would otherwise be possible.

Fetch has developers with individual experience spanning several decades in architecting and developing such systems and many others with detailed experience and knowledge in the field of distributed, highly parallel, scaleable networked applications in C++.

5.2 FULL AND THIN NODES

Fetch supplies the source and executables code for both full nodes and thin nodes. Thin nodes are a reduced version of the Fetch node that requires fewer resources. In particular, the memory, bandwidth and processing requirements are reduced on a thin node to allow for fully integrated node-AEA setups. The source code, executables and documentation will be made available on github.com. **A Fetch node is a class of AEA**: it has can send and receive Fetch tokens and its transactions are placed on the ledger with all other transactions.

5.3 NETWORK PARTICIPATION APP: NOT JUST A WALLET

Fetch is delivering a native wallet for desktop platforms (MacOS, Windows and Linux). For mobile platforms, the wallet app is additionally a *network participation application* (NPA). Fetch's NPA has a number of additional features over and above wallet functionality:

1. It **enables the user to create autonomous economic agents** to represent combinations of sensor information simply by dragging and dropping components. These unique arrangements of data and sensors can be configured to create unique AEAs that can then deliver that value to other agents in the Fetch world.
2. It **provides a *representative* agent**: one that represents you (see 2.5). These agents can take your preferences, as stored safely, securely and encrypted on your own personal device, and use that to provide tailored solutions to problems that you might have.
3. It **allows the user to explore the decentralised world** as though they were an agent — viewing the Fetch world and watching it changing rearranging itself in real-time.

The NPA is a key part of delivering Fetch's features and utility to individuals and providing the Fetch network with a large population of agents that are individually configured by users to represent unique combinations of sensor data, general information and human information.

5.4 AGENTS AND INTERFACES

The agent classes described in 2.5 are supplemented by pure interface AEAs. These are agents that exist as agents merely to interface to the OEF in order to extract data for analysis. This may then be transformed, combined with other proprietary data and made available specifically or generally for agents to then exploit in the world.

5.5 DATA PRIVACY AND PROTECTION

Fetch.AI's core proposition is to create a highly trusted environment in which Autonomous Economic Agents can exist, discover and be discovered, communicate with each other, broker and transact with the confidence that messages and transactions are both secure and between the parties intended. Privacy and Data Protection is of the highest importance to us. A key aspect of the Fetch protocol, the Smart Ledger and the supporting Distributed Ledger is that Personally Identifiable Information (PII) is not controlled or managed by Fetch. The only information required to transact is the public address of AEA. Given the AEA is the only holder of the private key involved, only the non-identifiable public portion is stored on the ledger and used to communicate. Agents are able to sign and decrypt themselves with the Fetch system facilitating.

Fetch.AI will provide market and transaction intelligence — interpreting one's individual transactions in context of all the aggregated transactions over the Fetch service. It will not be possible to reverse engineer this aggregated intelligence back into individual transactions.

6. FETCH: BIOGRAPHIES AND BACKGROUND

Fetch.AI is a convergent technology company founded in 2015 to leverage economic value generation by converging artificial life, machine intelligence, multi-agent simulations, cryptography and distributed ledger technologies.

With strong business, economic, software, regulatory and transport experience, it is ideally positioned to contribute to the future networked civilisation. Our core team includes:

HUMAYUN SHEIKH — CO-FOUNDER AND CEO, FETCH.AI

Humayun Sheikh, CEO. An innovation entrepreneur and founding investor in DeepMind with a record in revolutionising trading in the steel sector, he is now changing the way we transact and travel.

TOBY SIMPSON — CO-FOUNDER AND CTO, FETCH.AI

Producer of the successful a-life Creatures series of games and early developer at DeepMind, his thirty years' experience in software development and ten as a CTO are now focussed on crypto-economics.

THOMAS HAIN — CO-FOUNDER AND CHIEF SCIENTIFIC OFFICER

Professor at Sheffield and established scientist in advanced machine learning AI. He bridges the real world and academia and is inspired by the opportunities AI brings to modern society.

JONATHAN WARD — SENIOR MACHINE LEARNING SCIENTIST

Highly-cited researcher in AI excited by the potential of intelligent systems to create the economy of the future. PhD in machine learning from University College London with earlier training in physics and mathematics.

TROELS RØNNOW — SENIOR SOFTWARE ENGINEER

A scientist and innovator, he has benchmarked D-Wave Two, co-authored 35 patent applications and has been working more than two years full time building distributed ledgers.

JEROME MALOBERTI — MACHINE LEARNING DEVELOPER

Implementing large scale multi-agent systems with ML algorithms with nearly 20 years' experience following a first degree in Software Engineering and a PhD in Artificial Intelligence.

ARTHUR MEADOWS — COMMERCIALISATION AND MARKETING

International experience in software start-ups and bringing disruptive, high-growth tech products to commercial success. MBA from Judge Business School, University of Cambridge.

SEBASTIAN NICKEL — MACHINE LEARNING RESEARCH ENGINEER

Architecting ML models & algorithms for expressing individuals' actions and preferences. With a background in cognitive sciences & NLP, Sebastian holds degrees in both Mathematics and Psychology.

JOSHUA CROFT — DATA SCIENTIST

NLP and data mining specialist in geolocation, natural language and IoT. Ex-PlayStation developer with an MSc in Advanced Computer Science from the University of East Anglia.

NATHAN HUTTON — SOFTWARE ENGINEER

Experienced software engineer and FPGA specialist, previously worked at BAE on high speed communication systems. MEng from The University Of Edinburgh.

ROBERT DICKSON — SOFTWARE ENGINEER

A seasoned software engineer and architect with experience in dynamics simulations, low-level protocols, 3D graphics and virtual machines. He has a strong artificial life and mathematics background.

Peter Bukva — Software Engineer
Experienced software engineer and scientist. Previously worked at Bloomberg and Siemens Corporate Research. MSc in Solid State Physics and PhD in High Temperature Superconductors.

Ali Hosseini — Software Engineer
Expert on Multi-Agent Systems and Reasoning with a background in software engineering. PhD and MSc in Artificial Intelligence from King's College London with an excellent academic track record.

Katie Lucas — Senior Software Engineer
Ex-Google SRE and Privacy engineer, she has also worked with Citrix, Hitachi and Grapeshot. Highly experienced in many fields including security systems, military simulations and real-time data-processing pipelines

Edward FitzGerald — Senior Software Engineer
A engineer, researcher and technology enthusiast with experience of distributed ledgers, video codecs, networking and low level software optimisation. Strong engineering and mathematical background, he obtained his degree in Electronic Engineering from University of Surrey.

Gary Wood — IT and Security Manager
A multi-talented IT Manager with over 15 years experience. Well-versed at supporting fast moving, dynamic development teams. Previously designed, implemented and managed an onsite data-centre running a massively multi-player online game for the BBC.

Catherine Moriarty — Chief Amazement Officer
A customer service and experience expert, who spent many years with many businesses, driving efficiency, and intuitively fixing processes and procedures. Now bringing a sprinkle of amazing to the Fetch team.

Richard Dybowski – AI & ML Research Strategist
Senior AI research scientist and mathematician at the University of Cambridge, researching the application of AI/ML to medicine. PhD in the application of AI to computational chemistry from Leeds University.

Melvyn Weeks — Economic Advisor
Assistant Professor in Economics at University of Cambridge, researching the application of Machine Learning to market pricing. Senior Economic Advisor to Ofgem, UK's Energy regulator.

Monique Gangloff — Scientific/Biotech advisor
Principal investigator/senior scientist at the University of Cambridge, Department of Biochemistry. Her research endeavours have resulted in more than 35 international peer-reviewed publications and 1 patent application.

The authors would like to thank all those at Fetch and Outlier Ventures
for their valuable feedback and contribution to this document

https://fetch.ai
info@fetch.ai
https://t.me/fetch_ai

March 2018, May 2018