Peer-to-peer Autonomous Agent Communication Network

Lokman Rahmani
Fetch.ai
Cambridge, UK
lokman.rahmani@fetch.ai

David Minarsch
Fetch.ai
Cambridge, UK
david.minarsch@fetch.ai

Jonathan Ward
Fetch.ai
Cambridge, UK
jonathan.ward@fetch.ai

ABSTRACT

Reliable and secure communication between heterogeneously resourced autonomous agents controlled by competing stakeholders in a decentralized environment is a challenge. Agents require a means to find each other and communicate without reliance on a centralized party and participation in the system must be permissionless. We present the Agent Communication Network (ACN), a peer-to-peer lookup system that provides a distributed overlay to the Internet and addresses this problem. The ACN enables agents to find each other and to communicate safely. It achieves this by leveraging a distributed hash table for agent lookup, maintained by participating peers and through the use of public-key cryptography. The paper discusses the properties of the system and its guarantees as well as its integration with a novel multi-agent system. Preliminary benchmark results demonstrate the feasibility of the system, its performance, and its scalability.

KEYWORDS

peer-to-peer; distributed hash table; multi-agent communication

ACM Reference Format:
Lokman Rahmani, David Minarsch, and Jonathan Ward. 2021. Peer-to-peer Autonomous Agent Communication Network. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

1.1 Motivation

The dominant architecture in Web 2.0 is that of client-server in which the client is fully reliant on the server and the server determines whether a client is legitimate. This paradigm is a good fit for centralized platforms and applications.

With the rise of distributed ledger technology (DLT) multi-agent systems (MAS) are seeing renewed attention [1, 9, 34]. Unlike their historic counter-parts, modern agent and MAS implementations [29, 45] are conceived for multi-stakeholder environments and built for deployment at scale on the public Internet. For this to become a reality heterogeneous agents owned by competing entities need to be able to find and communicate with each other without reliance on a central party. In particular, the following problem needs to be addressed:

> given the identity of a target agent, how can the originator agent deliver a message to it whilst certain properties are guaranteed?

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

1.2 Requirements

Any point-to-point communication system between potentially competing stakeholders on the public Internet must satisfy a number of requirements to find adoption:

(1) Reliability: there need to be guarantees on message reception
(2) Authentication: it should not be possible for an entity to impersonate another entity
(3) Confidentiality: there should be no exposure of sensitive information about the message content to third parties
(4) Availability: the service should satisfy certain liveness guarantees

Authentication, Confidentiality and Availability together ensure (5) Security.

Furthermore, to support a multi-stakeholder multi-agent economy the following additional design constraints need to be satisfied:

(A) Distributed environment: agents should be able to locate anywhere in the public Internet
(B) Decentralized environment: agents should not require a centralized authority for communication and entry to the system must be permissionless
(C) Resource flexibility: agents should be flexibly deployable in differently resourced (i.e. CPU, memory, storage, network) devices

An implication of a decentralized environment is that by default there should be no requirement for agents to trust other system participants and agents should process messages asynchronously.

1.3 Contribution

The Agent Communication Network (ACN) addresses the problems of locating agents and establishing channels for message-based communication between them (as per section 1.1) in a distributed, decentralized MAS deployed on the public Internet. To satisfy (A) and (B), at its core the ACN operates in a peer-to-peer fashion where equally privileged participants known as peers collectively maintain a distributed overlay to the Internet and where entry to the system is permissionless. To satisfy (C), agents can participate in the ACN in different capacities: operating their own peer or as a client relying on an existing peer.

The distributed overlay is implemented on top of a *distributed hash table* (DHT) [39, 44]. Similar to regular hash tables, a DHT stores key-value pairs. However, it does so across multiple peers. Pairs assignment to peers is decided using its *consistent hashing* algorithm. For efficient peer routing and pair retrieval, the peers self-organize following a specific topology as a logical layer on top of a network one, constructing a *structured overlay*. Peers' network locations are maintained in local routing tables. In the case of the ACN, the DHT is used to store the associations between an agent's id and its peer's id as key-value pairs. The agent id is the agent's

address (a hash of its public key) and the peer id is a multihash [23] of the peer's public key.

To be reachable in the ACN, an agent has to register itself to the DHT through a participating peer, either already existing or deployed for the occasion. The peer acts as the agent contact in the ACN and delivers its incoming messages from other agents. Likewise, to reach another agent, the initiator agent generates the message and forwards it to its contact peer which performs a lookup in the DHT for the corresponding contact peer of the destination agent. If found, the contact peer of the sending agent opens a communication channel and delivers the message to the corresponding peer.

The ACN relies on public-key cryptography [16, 41] as an integral part of its design. In a trustless environment, public-key cryptography offers guarantees on communication such as authentication (2), confidentiality (3) and integrity (hence availability (4)). In the ACN public keys (and addresses derived from them) are used as identities for both peers and agents. Key agreement protocols are used for peers to authenticate each other but also for agents to authenticate their contact peers. Attaching digital signatures to the stored DHT records guarantees their authenticity, allowing any participant in the system to verify the association between a given peer and agent. Both yield confidentiality where no peer or agent can receive messages of other peers and agents and all communication is end-to-end encrypted.

Peers and agents are by design two separate entities in order to obtain (I) separation of concerns conceptually and (II) flexibility of deployment practically. Agents' execution environment can be resources-constrained with limited memory space, CPU power, and network bandwidth. To accommodate for that, the ACN offers different types of connections that an agent can have to its contact peer, each with a varying distribution of roles and workloads. Other common components of MAS such as service discovery, agent implementation and agent communication protocols are delegated to services and frameworks which utilise the ACN.[1]

The ACN is designed to be used with different transport layers and independently guarantees service reliability (1). In practice, TCP is the dominant layer however none of the primitives prevent utilisation of other transport protocols (e.g. UDP or Bluetooth etc.).

2 ARCHITECTURE

The communication system comprises a peer-to-peer (p2p) network running the DHT, a set of connections linking agents to peers, and a set of ACN-specific protocols prescribing interactions between agents and peers for correct delivery of messages.

2.1 Preliminaries

Identity. Each peer and each agent has a unique identity (id). The peer id is a public key or a (usually compact) representation of it. The agent id is an address derived from a public key via a hashing function. The representation should be unique to each public key and their association can be independently verified. We assume that the identified entity is the sole holder of the private key corresponding to the public key used as its identity.

Message. A message consists of metadata and payload.[2] The payload is the content of the message, i.e. encoded data (e.g. [13]). Metadata are information related to the message. Within the ACN, for each message, a metadata section includes its originator id, its destination id, and information related to payload encoding. In practice we make use of more metadata as discussed in section 2.7.

2.2 Peer-to-peer network

Peers are the core entities of the system. They are functionally equivalent and collectively constitute the peer-to-peer network. They run a distributed name service for agents and act as communication proxies for them. The name service is used to retrieve the contact peer of any previously registered agent. It is enabled by a distributed hash table (DHT) maintained by every peer.

2.2.1 DHT. A DHT is a peer-to-peer system that provides a lookup service similar to a regular hash table. However, the DHT stores the key-value pairs across multiple peers.

Consistent hashing. To assign responsibility for storage among peers, the DHT uses *consistent hashing* [20, 44]. A consistent hashing scheme uniquely maps record keys - the agents' addresses - and peers' ids alike to a virtual space. This produces a set of key identifiers and a set of peer identifiers respectively. The virtual space is then partitioned between peers where a record is stored by the peer whose identifier is the closest to the record's key identifier, following a distance metric defined in the virtual space. Consistent hashing has the property of requiring only a limited movement of keys when peers join or leave the network. This feature allows to develop solutions that are robust under changing environments and imperfect information. The consistent hashing mapping is also uniform thus each peer stores roughly the same number of keys. This tends to balance the load over the peers thus reducing hot spots in the system.

Construction. DHTs are constructed incrementally. The first peer, also called genesis, defines the virtual space and the mapping (hash) function and initially is responsible for the entirety of the space. New peers join the DHT by connecting to at least one of the existing peers. The new peer then announces itself to the network and receives from close peers the records that fit within its new responsibility zone. Different peer discovery mechanisms (the process of joining the network and finding other peers) can be implemented. The genesis peer can be hard coded in the distributed software or provided as an argument on launch and acquired via a third-party service.

Structured Overlay. Since peers only store a subset of all items, lookups need to be routed whenever a peer receives one for a key that it is not responsible for. Hence, each peer has to maintain a local routing table that contains other peers' identifiers and contact details in the form of a physical network address (TCP/IP address). The global graph created by connectivity between peers is called an overlay network. A peer's routing table defines its neighbors in the connectivity graph. For efficient routing, DHTs use specific algorithms to maintain a *structured overlay* [2] where the links define

[1] Finding other agents in the ACN is not aimed at service discovery, rather it is a necessary side-effect of DHT routing algorithms and only serves message delivery.

[2] [29] distinguish further between Envelope and Message. Here, we omit this distinction for simplicity.

a topology such as a tree, ring, torus, or similar. DHTs leverage the structure of the connectivity graph to guarantee efficient lookup operation in number of steps, for any key in the system.

Lookup operation. DHTs employ a recursive algorithm for record lookup. The lookup initiator starts by selecting from its routing table the closest peers to the record key. The initiator then forwards the lookup request to the peers it has chosen. Upon the reception of the request, a peer either returns the corresponding record in case it stores it, or a set of peers that are the closest to the record key. In the recursive step, the initiator re-sends the request to peers it has learned about from previous interactions. The lookup procedure halts when any peer returns the value or if no more closest peers are learned about.

2.2.2 Kademlia. Kademlia [28] with its secured addition S/Kademlia [3] is used for the DHT implementation. S/Kademlia uses 160-bits as its virtual identifier space and binary XOR as its distance metric. It uses cryptographic hash functions such as SHA1 to map peer identity and record keys to the identifier space. In S/Kademlia, the peers' connectivity graph takes the form of a binary tree, which combined with the XOR distance metric ensure efficient routing. DHTs in general and S/Kademlia specifically have the following properties:

- Decentralization: DHTs are fully distributed: no node is more important than any other. This improves robustness and makes DHTs very appropriate for multi-stakeholders environments.
- Scalability: Kademlia lookup grows as the log of the number of peers, so even very large systems are feasible. Caching is also possible for hot records.
- Load balance: DHTs act as a distributed hash function, spreading keys evenly over the peers through consistent hashing. This provides a degree of natural load balancing.
- Availability and fault-tolerance: Kademlia offers high availability through its self-stabilization protocols and correction-on-use properties. Kademlia also replicates each record by storing it over k closest peers, where k is a system-wide parameter. Self stabilization maintenance protocols automatically adjusts peers internal routing tables to reflect newly joined peers as well as peers failures, ensuring that, barring major failures in the underlying network, a previously stored record can always be retrieved. Correction-on-use take advantage of lookup and join operations to proactively correct a peers view of the system.
- Resilience: S/Kademlia is resilient to most common malicious attacks on the system. Kademlia naturally provides resistance to Denial-of-Service attacks by not relying on a central entity. S/Kademlia provides additional resistance to Eclipse attacks [42] and Sybil attacks [5].

2.3 Protocols

ACN protocols augment the peer-to-peer network to suite agents' communication needs and requirements. They leverage the DHT by storing agents' contact information and define rules for interactions between participants. Agent registration, de-registration and lookup are the main protocols.

2.3.1 Agent registration protocol. To be able to use the system, an agent has to register itself first. It requires as a preliminary the information of at least one peer participating in the system. Namely, the peer's URI (host and port number) to open a communication channel and its identity to authenticate it. After establishing the connection, the agent first generates a signature of the contact peer's id and sends it to its peer. The signature acts as a proof of representation for other peers in the peer-to-peer network. The peer then advertises the association between itself and the agent in the DHT as a (key-value) record, thus becoming the agent's official *contact peer*.

The peer adds the agent to its list of agents and listens for outgoing message from the agent side and incoming messages from the DHT side.

2.3.2 Agent lookup protocol. To send a message, an agent has to construct it and forward it to its contact peer.

A peer keeps listening for messages from all its registered agents. Once it receives a message it checks its destination field. If the destination agent is registered within the same peer it delivers it directly to it. If not it has to lookup the contact peer of the destination agent (see section 2.2.1). If a record is registered for the destination agent the peer opens a direct communication channel to the destination peer and requests its proof of representation. The peer uses the destination agent id to verify the authenticity of the proof and upon success marshals the message to the destination contact peer. The destination contact peer delivers the message to its agent. If no record is available for the destination agent the contact peer returns an error message.

2.3.3 Agent de-registration protocol. Currently, we do not support explicit agent de-registration. Instead, we rely on a Kademlia feature where each stored record has a timeout which when reached means the record is deleted from peers. To keep a record available, the initial peer who requested its storage needs to republish it periodically before the timeout.

2.4 Connections

Connections enable agents to use the peer-to-peer network. They implement the agent's side of the communication as clients and rely on corresponding services run on the peers as servers.

We propose different types of connections - a multi-tier architecture - to accommodate for agents' different requirements on system resources (CPU, memory, storage, network), confidentiality and level of trust in the contact peer. Further discussion about the possible trade-offs between these requirements and their advantages and disadvantages is provided in section 2.5.

2.4.1 Direct connection. This connection assumes that the agent and the peer are running on the same machine, and that the agent stays connected for its full service. This is for the case where an agent wants to deploy its own contact peer for full control. The peer participates in maintaining the DHT and must be reachable from the Internet. The agent can control which services to enable on the peer. The connection uses inter-process communication (IPC) mechanisms (i.e. named pipes) to efficiently exchange messages between the agent and the peer. Figure 1 illustrates an agent using a direct connection to a peer.

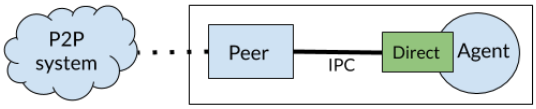

Figure 1: Agent using a Direct Connection. Agent and peer are located on the same machine and the connection relies on named pipes for communication.

2.4.2 Relayed connection. Similar to the direct connection, the agent and the peer are assumed to be running on the same machine. However, the contact peer is not publicly reachable from the Internet, most commonly due to restrictions imposed by network address translation (NAT). Thus it requires the relay service of another peer. However, it does not require its trust as the communication with a third peer can be end-to-end encrypted. This results in a double encryption: the communication between the origin and relay, the relay and target as well as the origin and target is encrypted. This connection is needed for the case when an agent wants full control over the peer but does not have a public URI, although it sacrifices some confidentiality as the relaying peer knows which agents and peers it is communicating with and the communication pattern. Figure 2 illustrates an agent using a Relayed connection to its contact peer.

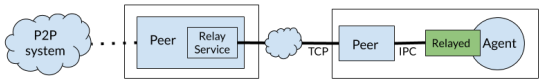

Figure 2: Agent using a Relayed connection. Both agent and relayed peer are on the same machine. When a peer is run in relayed mode it requests the relay service of one of its entry peers. If granted, the relayed peer's presence in the DHT and all its communication go through the relay peer.

2.4.3 Delegate connection. The Delegate connection allows an agent to use a remote peer as its contact peer in the ACN. This is possible through the delegate service that the contact peer can host. Communication between the agent and its contact peer is done using secured TCP connections. This connection requires the agent to trust the peer as it has access to the agents un-encrypted messages. This approach is most useful for agents on small devices which can still maintain a TCP connection. The trust requirement can be relaxed at the cost of performance where the agent can request its target agent to run a key agreement, or simply use asymmetric encryption. Refer to subsection 2.5 for further discussion. Figure 3 illustrates an agent using a Delegate connection to its contact peer.

2.4.4 Mailbox connection. All previous connections require the agent to keep the connection open to receive messages destined to it. Failure to do so returns an error to the message initiator by the agent's contact peer. The Mailbox connection relaxes this constraint by using a mailbox service offered by a contact peer. If a message is received for an agent registered with the mailbox service, the

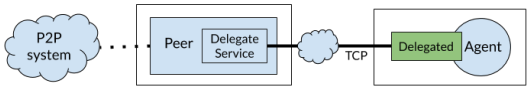

Figure 3: Agent using a Delegate connection. The peer offering the service can be operating by the same entity managing the agent but can also be different.

peers store it in a queue.[3] At a later stage the agent can query the peer for messages through HTTP requests.[4] This connection is useful for agents running on resource-constrained devices and that cannot maintain an open connection with the contact peer. Figure 4 illustrates an agent using a Mailbox connection to its contact peer.

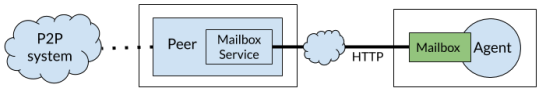

Figure 4: Agent using a Mailbox connection. The agent communicates with its contact peer through HTTP requests. The agent does not maintain an active connection to the peer, it connects only for the time to achieve the operation it intends to perform.

2.4.5 Typical deployments. To clarify the usage of the proposed connections, figure 5 illustrates examples of a typical deployment of the different connections, the peer-to-peer network, and the ACN as a whole with participating organization boundaries. By design, there can be multiple disjoint ACNs. This is the case when nodes bootstrap against a disjoint set of genesis nodes. To establish a common public network, a set of public peers should be always running and their addresses accessible (e.g., P1, P2 and P3 in the figure). Other peers subsequently join through these peers. Similarly, private organizations which want to be part of the network can do so as well by deploying their own peers (e.g., Peer A and Peer B in the figure). Each peer can decide which service to enable depending on available resources and the incentives to do so. In the network described in the figure, P3 has the relay service enabled and is serving P4 for agent PR1. On the other hand, P2 has only delegate and mailbox services enabled and is serving agents PD1 and PM1. A peer can perfectly disable all the services and only run DHT operations (e.g, P1). Peers can also use an authorization list for agents allowed to use its services. In the presented network, Peer A and B although publicly accessible, serve only agents from within their respective organization. Importantly, with all these different configurations and potential restrictions, every agent is still able to find every other agent and communicate with it securely.

[3] Similar to the Delegate connection, the Mailbox connection requires that the agent trusts the peer. However for this connection it can be alleviated only through asymmetric encryption as key agreement would be very costly or not possible at all. Refer to subsection 2.5 for further discussion.

[4] For now, it is assumed mailboxes are centralized, in that they are maintained by one and only one peer. If the peer goes offline or has a fault the mailbox's contents are lost. In the future, a decentralized mailbox service could be implemented where more than one peer maintains the same mailbox. Thanks to the modular setup this can be retrofitted.

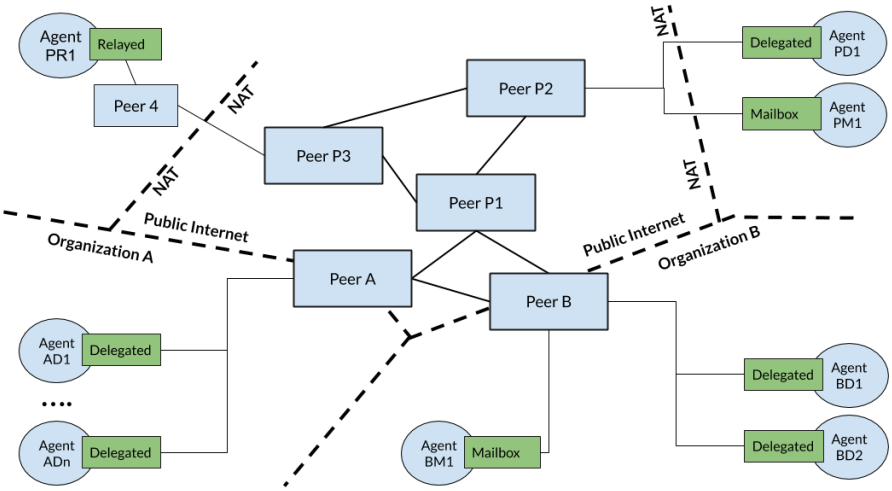

Figure 5: Typical ACN deployment. ACN allows for flexible deployment of agent and peers considering their resources and affiliations while ensuring that every agent can find every other agent and communicate securely.

2.5 Security, Trust & Performance

This section provides further details about security properties of connections and their impact on performance requirements.

2.5.1 End-to-end encryption. All point-to-point communication between entities (peers and agents) is end-to-end encrypted using TLS [40] with pre-shared public keys. This guarantees authentication, confidentiality and message integrity security properties. Unlike previous work [12, 18], the ACN does not require certification authorities to distribute public keys. Instead, public keys are pre-shared in the form of addresses, obtained via hashing the public key. Peers' addresses are distributed as part of their contact information along with their IP addresses and TCP ports. Agents' addresses are included in the message metadata as its source or destination. Once a connection is initiated between two entities, they start by exchanging their public keys. Public keys are verified by computing their hashes and comparing the resulting addresses to the pre-shared ones. If successful, they proceed to TLS handshake.

Consequently, the above is sufficient to ensure security properties in a trustless environment for communication between peers and communication between agent and peer via Direct connection. The former is true because DHT operations do not require any sensitive information to be exchanged between peers. The later is true as well because both agent and peer are owned by the same authority. However, communication via the rest of agents' connections needs more consideration to ensure all security properties are met without requiring the trust of the (third-party) intermediary peer they rely on. This is the case because for Relayed, Delegate, and Mailbox connections messages are decrypted as they go through the intermediary peer. To that end, a second TLS channel needs to be established between communicating parties leading to double encryption. Double encryption is costly and may not be possible if a session can not be maintained, i.e., in the case of the Mailbox connection. A trade-off between performance and security properties is to use asymmetric encryption which ensures only confidentiality. Figure 6 shows a diagram of communication channels annotated with the security protocols they use.

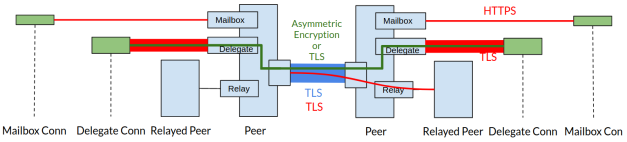

Figure 6: All point-to-point communication protocols use public-key cryptography to ensure security (authentication, confidentiality, and availability) using TLS handshakes with pre-shared public keys. The figure shows the different connection configurations and their security model.

2.5.2 Trust. The relationship between the contact peer and the agent is analogous to a server-client relationship, in particular it is fully trusted. Agents need to trust peers via which they connect to provide them a true representation of the DHT. However, this is not a limiting factor since either the peer is operated by the same entity as the agent or the peers compete to provide services to agents and agents can detect when they are being cheated by their contact peer (as it is not possible for it to forge incoming our outgoing agent messages).

2.6 Economic model

The operation of the DHT incurs costs (CPU, memory, storage and networking) for the peers maintaining it. This section discusses the incentives peers have to participate in its provision as well as the incentives of agents which do not run a contact peer.

2.6.1 Incentives for participation. By design, agents require a contact peer to use the ACN for communication with other peers and

agents. In the absence of alternatives, this incentivises every participant to deploy its own peer and help maintain the DHT. Peers are not charged for participation in the peer-to-peer network as they contribute to its maintenance through their participation alone.

For agents that do not have the means to deploy a peer or choose not to, they can either rely on a third party organization to deploy one on their behalf or use a share of an already deployed peer.

In its current design, the peer-to-peer network does not pose any restrictions on the number of agents a peer can represent in the DHT. However, agents who do not maintain their own peer impose an externality on the peers maintaining the DHT. In particular, they impose storage requirements. Due to the properties of the DHT these are uniformly born by all the peers. There are no externalities from networking on third party peers as all traffic is incurred by the contact peers of the agent and its target.[5] The networking costs for the agents' contact peer imposes a direct constraint on the number of agents a peer can represent in the DHT.

A contact peer can charge its agents for access to the network using a pricing model of its choosing.[6] Since the peers offering Delegate or Mailbox connections compete with each other in providing services to the agents, and since anyone can become a peer, we expect the service offered by the peers to be priced competitively and potentially be free. We also expect, for the same reasons, that the pricing problem faced by peers can be studied largely by taking the other system components as given.

2.7 Beyond message delivery

As discussed, the communication network addresses agent id-based message delivery only. To fully address the communication problem, the agent implementation is required to handle the rest. To that effect, this work utilises a novel agent framework [29]. The framework's communication system offers multiplexing of connections from different sources and protocols to guide the exchange of messages as dialogues from start to end. Protocols can be user-defined and generated. The framework is also responsible for setting up and tearing down the connections as well as configuring them.

The ACN does not address the need of generic service discovery in a MAS. This can be offered by agents or external services deployed on top of or alongside the ACN following proposals in the literature [15, 19, 43].

2.8 Related Literature

Peer-to-peer systems are relatively well researched and have seen growing attention in recent years [11, 25, 28], in particular with the rise of crypto currencies [8, 31]. The literature acknowledges the natural connection between peer-to-peer systems and MAS [9, 21, 37, 45], in particular in multi-stakeholder environments.

Through contributions (I) and (II) (cf. section 2.4) the design of the ACN is agent framework independent (although its usage is exemplified utilising a specific framework [29]) and hence distinct to monolithic systems as proposed in [26, 27, 32, 35, 43]. The ACN allows for point-to-point communication between heterogeneously resourced agents and different agent types and implementations [4, 6, 29, 49]. In particular, any message-based agent communication language is supported (e.g. FIPA ACL [13]).

We consider high performance as a first class criterion for our system design and implementation to be used at Internet scale. In addition to the aforementioned, this includes, (i) usage of a structured overlay (unlike solutions like [26] based on unstructured ones), (ii) lookup request routing rather than message relaying [27, 35], (iii) usage of binary format serialization for wire transfer rather than XML ones [32, 35], and (iv) an efficient implementation utilising [36].

Building trust and reputation [33, 50] as well as fostering cooperation [24, 30] are well-studied topics in peer-to-peer systems and the ACN lends itself towards extension in these directions.

Kademlia is known to be quite stable against DoS attacks and some form of Sybil attacks. There is a rich body of research which shows how to make it more resilient [3, 14, 30, 33, 47].

3 BENCHMARK

The ACN as described is fully implemented except for the Mailbox connection. A production version is deployed on the public Internet and exposed to regular use.

In this section, we demonstrate the feasibility of the system and its performance by benchmarking its main operations on a reference implementation that excludes certain security features, namely the signing and verification of a peer's proof-of-representation when registering the agent and when looking it up, respectively. The following first describes the reference implementation of the ACN and benchmarking setup. Then, it presents preliminary benchmark results of the ACN operations and finally discusses them. We measure the following operations:

- *Agent registration*: the time for a peer to register a new agent address to the network.
- *Agent lookup*: the time for a peer to get the contact peer of a given agent address.
- *Peer join*: the time for a new peer to start and successfully register its agent.
- *Peer echo*: the time for a peer to lookup the contact agent of a given agent address and exchange one message with it, achieving one round trip.

For reference, we also provide measurement of simple RTT (round trip time) on non-secure TCP connections as a baseline for communication performance on the benchmark machine.

3.1 Implementation details

In our reference implementation, peers are implemented in Golang as nodes and connections are implemented in the Python programming language as components to an autonomous agent framework [17, 29]. Golang facilitates writing efficient and scalable concurrent programs thanks to its Goroutines and channels built-in features. The peers use the go-libp2p [36] implementation of S/Kademlia DHT. The go-libp2p library has been proven as part of the running IPFS network [22]. The public-private key pairs are generated from the elliptic curve as specified by the standard SECP256k1 [7].

[5] The case of the relay peer is an exception. However, a relay peer is free to ignore traffic.

[6] Peers and agents can, for instance, utilise smart contracts [48] deployed on a DLT to manage payment and staking of funds.

3.2 Setup details

To eliminate network latency variability, all the benchmarks are run locally on an AMD64 Linux machine with 12-cores Intel Core i7-8750H 2.20GHz CPU and 16GB of RAM. Other tests have been conducted over the Internet but are not reported here.

Each benchmark (cf. [38]) is setup by deploying an ACN network of a given size in number of peers. It is constructed iteratively where each new peer is given the previous one as entry to the network. Once the network is functional, a given operation is executed and its run time is measured. Measures are repeated at least 20 times and each benchmark is run 20 times. Varying the size of the network allows us to test the scalability of the selected operation.

3.3 Results

Figures 7 and 8 present the benchmark results of the defined operations, grouped into atomic (agent registration and lookup) and end-to-end (peer join and echo) classes. For both figures, the x-axis shows the size of the network in number of peers and the y-axis shows the operation execution time in milliseconds. Both axes are in logarithmic scale. Each plot is labeled with the operation it measures. Table 1 reports the same measures with standard deviation.

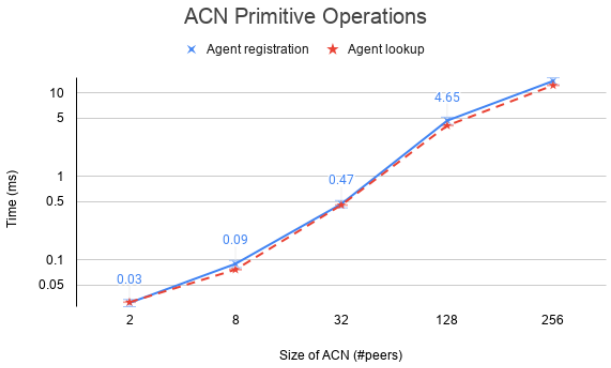

Figure 7: ACN primitive operations performance in function of network size. TCP RTT is 0.040ms.

3.4 Discussion

From figure 7, we can see that for small sized networks of 2 to 8 peers the execution cost of atomic operations is the same or very close to the TCP RTT baseline. This demonstrates a very low to non-existent overhead for the peers' point-to-point communication implementation. We can also observe that both agent registration and lookup operations perform similarly. This is expected as they use the same routing algorithm where one stores the record and the other fetches it. End-to-end operations follow the same trend as their atomic counterpart, except for peer join operations where its cost is order of magnitudes higher. This reflects the overhead of peer setup and is most noticeable for small size networks as it gets amortized very quickly with larger networks (e.g., entirely amortized within networks of 256 peers). All benchmarked operations scale well, although linearly not logarithmically. This could be

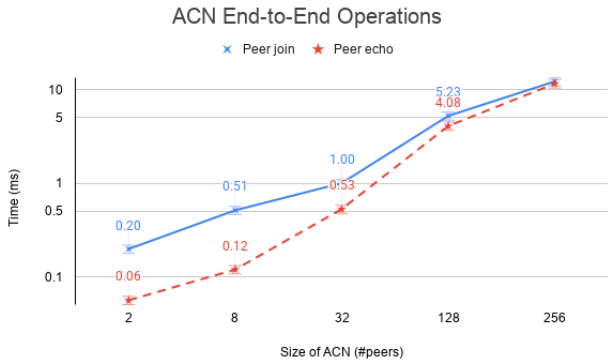

Figure 8: ACN end-to-end operations performance in function of network size. TCP RTT is 0.040ms.

explained by the benchmark setup running on a local machine with very low network latencies thus shifting most of the operation cost on computations. In intended deployments of the ACN over the Internet, network latency is magnitudes higher making the number of peers to contact the major part of the operation execution time. Tuning system configurations is also expected to improve its scalability, as benchmarks were run with the default configuration.

Above results demonstrate the low execution cost of the system and its scalability, thanks to go-libp2p's efficient implementation of the DHT and minimal overhead of the ACN protocols' design and implementation. They demonstrate the system's feasibility and efficiency. Further tests are needed to practically demonstrate system properties described in this paper with representative deployment, especially at larger scale and under faulty conditions.

4 DISCUSSION

The previous sections present in detail the ACN, its architecture, its components, and initial benchmark results using its reference implementation. In this section, we elaborate more on topics and questions that arise from using a DHT as a communication system for MAS in a distributed decentralized environment.

4.1 Resiliency and fault tolerance (DHT)

The S/Kademlia DHT used by the ACN is tolerant to faults and resistant to most malicious attacks. If a peer fails or disconnects, after a grace period its neighbours simply remove it from their routing tables and query the network to update their routing table with the next closest peer, effectively bypassing the failed one. Records stored at the failed peer are not lost as they are already replicated to its neighbours. For the stability of the network, peers always privilege long standing ones as neighbours. This also partially mitigates Sybil attacks where an entity joins the DHT with multiple peers and uses them to gain disproportional influence in the system that it exploits to its advantage. Because of the use of cryptographic hash functions for the consistent hashing scheme, it is very hard to orchestrate a DoS attack where a specific peer is targeted, as this would require reversing the hash function to produce keys that have hashes close to the targeted one. Similarly,

Table 1: ACN primitive and end-to-end operations benchmark results. Measures are repeated at least 20 times each and each benchmark is run 20 times.

#peers	2	8	32	128	256
Agent registration	0.03 ± 0.01	0.09 ± 0.01	0.47 ± 0.16	4.65 ± 3.31	13.84 ± 6.9
Agent lookup	0.03 ± 0.01	0.08 ± 0.01	0.45 ± 0.16	4.05 ± 0.76	12.27 ± 4.33
Peer join	0.20 ± 0.04	0.51 ± 0.26	1.00 ± 0.08	5.23 ± 2.03	12.22 ± 1.4
Peer lookup	0.06 ± 0.02	0.12 ± 0.06	0.53 ± 0.11	4.08 ± 0.41	11.57 ± 1.33

it is very hard for an attacker to isolate a target peer by generating peer ids with hashes close to it. This is known as eclipse attack. Finally, to limit spamming the network, peers only store *pointers* to the DHT records, i.e., only the key of the record and the id of the peer requesting its storage.

4.2 Name service only vs. message routing

The ACN provides an agent address-based lookup service that is functionally similar to the Internet's DNS but distributed and decentralized. Agent lookup requests are routed throughout the DHT by peers to where its registration record is stored. Once the record is retrieved, a direct connection is established between contact peers of the initiator and target agents. This means that we do not route the message itself but only the request to find the target agent, as the message is destined to one agent only in the system. This naturally consumes less network traffic and better protects communication confidentiality.

4.3 DHT vs DLT

As part of a multi-stakeholder MAS which includes DLT, the ACN must be clearly distinguished from the latter set of technologies. DLT systems operate under the premise that a consensus is to be reached amongst participants of the systems on some state of the system at regular intervals. The ACN, does not attempt to establish consensus about the contents of the DHT at any point. On the contrary, it is by design that no global state is observed by any of the system participants. This design feature contributes to the relative scalability of the ACN in comparison to DLTs.

4.4 Agent mobility

To use the ACN, each agent must associate with a peer participating in it. This does not prevent agents from switching between contact peers. Changing to a new contact peer simply involves deregistering from the current one and registering with the new one. The old peer may still receive requests to receive messages for the agent until the old record has been removed from the DHT. Honest peers decline the message and the agent can enforce this by adding an expiration date to its contact peer's proof-of-representation.

4.5 Free-rider problems

As discussed in section 2.6, peers can impose a negative externality on each other by registering an excess amount of agents. There, we argue that the design of the ACN naturally dampens some of these problems. In the future, we might choose to modify the Kademlia implementation to preempt these problems [30]. Alternatively, or in addition we can also investigate incorporating smart contracts deployed on a public DLT to regulate access to the ACN whilst maintaining its permissionless entry feature.

4.6 Limits to point-to-point message-based communication

The ACN is designed around a bilateral message-based interaction paradigm. As such, message broad-casting and flooding is not supported as a primitive. However, multi-lateral interactions, including broadcasting to a set of known agents and gossiping [46], can be implemented utilising the bilateral primitives. Feed-based interactions are equally implementable as a higher level abstraction.

4.7 Advanced Search and Discovery

In principle, a DHT can be used to store arbitrary data. An example of such usage of a DHT is the IPFS system [22]. In the ACN, the use of the DHT is limited to recording associations between agent and peer ids and service discovery is assumed to be provided a separate service.

For some use-cases it might be useful to express a finite number of agent types on the DHT level and therefore add a further association between agent id and agent type to the DHT. This functionality would potentially be useful to provide a simplistic discovery service. However, it is also open to misuse as agent types cannot be independently verified.

4.8 Permissioned usage

In principle, the ACN can be operated in a permissioned manner if peers are endowed with access lists, for instance. Such a permissioned network can serve managed agent economies.

5 CONCLUSION

This work introduces a self-contained system for address-based agent lookup using a distributed hash table. The system is operated by peers in a decentralized manner with permissionless entry. Trust and security are enabled by using public cryptography (TLS/HTTPS with pre-shared public keys). The solution allows for flexible deployment and control over available resources, using a multi-tier architecture. It is suitable for use with heterogeneous agent implementations and applications [9, 10, 29, 34]. The benchmark demonstrates that the current implementation is performant and scalable and that the system design is feasible. At the time of publication, the system is deployed on the public Internet and continuously used by several thousand agents in production. Preliminary discussions confirm a largely incentive compatible design. However, further work needs to be undertaken to ensure the incentives of system participants are aligned in all cases.

REFERENCES

[1] Ilya Afanasyev, Alexander Kolotov, Ruslan Rezin, Konstantin Danilov, Manuel Mazzara, Subham Chakraborty, Alexey Kashevnik, Andrey Chechulin, Aleksandr Kapitonov, Vladimir Jotsov, Andon Topalov, Nikola Shakev, and Sevil Ahmed. 2019. Towards Blockchain-based Multi-Agent Robotic Systems: Analysis, Classification and Applications. arXiv:1907.07433 [cs.RO]

[2] Luc Onana Alima, Ali Ghodsi, and Seif Haridi. 2004. A Framework for Structured Peer-to-Peer Overlay Networks. In *IST/FET International Workshop on Global Computing - Volume 3267*. Springer-Verlag, Berlin, Heidelberg, 223–249.

[3] Ingmar Baumgart and Sebastian Mies. 2007. S/kademlia: A practicable approach towards secure key-based routing. In *2007 International Conference on Parallel and Distributed Systems*. IEEE, 1–8.

[4] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. 2007. *Developing multi-agent systems with JADE*. John Wiley & Sons.

[5] Arpita M Bhise and Shailesh D Kamble. 2016. Review on detection and mitigation of Sybil attack in the network. *Procedia Computer Science* 78 (2016), 395–401.

[6] Rafael H Bordini, Jomi Fred Hübner, and Michael Wooldridge. 2007. *Programming multi-agent systems in AgentSpeak using Jason*. John Wiley & Sons.

[7] Daniel RL Brown. 2010. Sec 2: Recommended elliptic curve domain parameters. *Standars for Efficient Cryptography* (2010).

[8] Vitalik Buterin. 2013. Ethereum Whitepaper. https://ethereum.org

[9] Davide Calvaresi, Alevtina Dubovitskaya, Jean Paul Calbimonte, Kuldar Taveter, and Michael Schumacher. 2018. Multi-Agent Systems and Blockchain: Results from a Systematic Literature Review. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection*, Yves Demazeau, Bo An, Javier Bajo, and Antonio Fernández-Caballero (Eds.). Springer International Publishing, Cham, 110–126.

[10] Davide Calvaresi, Mauro Marinoni, Arnon Sturm, Michael Schumacher, and Giorgio Buttazzo. 2017. The challenge of real-time multi-agent systems for enabling IoT and CPS. In *Proceedings of the international conference on web intelligence*. Association for Computing Machinery, 356–364.

[11] NM Mosharaf Kabir Chowdhury and Raouf Boutaba. 2010. A survey of network virtualization. *Computer Networks* 54, 5 (2010), 862–876.

[12] Joris Claessens, Bart Preneel, and Joos Vandewalle. 2001. Secure communication for secure agent-based electronic commerce applications. In *E-Commerce Agents*. Springer, 180–190.

[13] IEEE FIPA Standards Committee. 2001. *Communicative Act Library Specification*. Technical Report. Foundation for Intelligent Physical Agents.

[14] Zoltán Czirkos and Gábor Hosszú. 2010. Enhancing the Kademlia P2P Network. *Periodica Polytechnica Electrical Engineering (Archives)* 54, 3-4 (2010), 87–92.

[15] Elena Del Val, Miguel Rebollo, and Vicente Botti. 2014. Enhancing decentralized service discovery in open service-oriented multi-agent systems. *Autonomous agents and multi-agent systems* 28, 1 (2014), 1–30.

[16] Whitfield Diffie and Martin Hellman. 1976. New directions in cryptography. *IEEE transactions on Information Theory* 22, 6 (1976), 644–654.

[17] Marco Favorito, David Minarsch, Ali Hosseini, Aristotelis Triantafyllidis, Diarmid Campbell, Oleg Panasevych, Kevin Chen, Yuri Turchenkov, Lokman Rahmani, and Jiří Vestfál. 2019. Autonomous Economic Agent (AEA) Framework.

[18] Qi He and Katia Sycara. 1998. Towards a secure agent society. *ACM AA* 98 (1998).

[19] Rajaraman Kanagasabai et al. 2013. Semantic Web service discovery: state-of-the-art and research challenges. *Personal and ubiquitous computing* 17, 8 (2013), 1741–1752.

[20] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. 1997. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* (El Paso, Texas, United States) *(STOC '97)*. ACM, New York, NY, USA, 654–663. https://doi.org/10.1145/258533.258660

[21] Manolis Koubarakis. 2003. Multi-agent systems and peer-to-peer computing: Methods, systems, and challenges. In *International Workshop on Cooperative Information Agents*. Springer, 46–61.

[22] Protocol Labs. [n.d.]. *IPFS*. https://ipfs.io/

[23] Protocol Labs. [n.d.]. *Multihash*. https://multiformats.io/multihash/

[24] Kevin Lai, Michal Feldman, Ion Stoica, and John Chuang. 2003. Incentives for cooperation in peer-to-peer networks. In *Workshop on economics of peer-to-peer systems*. 1243–1248.

[25] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. 2005. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials* 7, 2 (2005), 72–93.

[26] Ziyan Maraikar. 2006. Resource and service discovery for mobile agent platforms. *Master's thesis, Department of Computer Science, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands* (2006).

[27] Marco Mari, Agostino Poggi, Michele Tomaiuolo, and Paola Turci. 2005. Enhancing multi-agent systems with peer-to-peer and service-oriented technologies. In *In proc. of 6th International Workshop From Agent Theory to Agent Implementation (AAMAS)*.

[28] Petar Maymounkov and David Mazieres. 2002. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*. Springer, 53–65.

[29] D. Minarsch, S. A. Hosseini, M. Favorito, and J. Ward. 2020. Autonomous Economic Agents as a Second Layer Technology for Blockchains: Framework Introduction and Use-Case Demonstration. In *2020 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 27–35. https://doi.org/10.1109/CVCBT50464.2020.00007

[30] Tim Moreton and Andrew Twigg. 2003. Enforcing collaboration in peer-to-peer routing services. In *International Conference on Trust Management*. Springer, 255–270.

[31] Satoshi Nakamoto. 2019. *Bitcoin: A peer-to-peer electronic cash system*. Technical Report. Manubot.

[32] B. J. Overeinder, E. Posthumus, and F. M. T. Brazier. 2002. Integrating peer-to-peer networking and computing in the AgentScape framework. In *Proceedings. Second International Conference on Peer-to-Peer Computing,*. IEEE, 96–103. https://doi.org/10.1109/PTP.2002.1046318

[33] Riccardo Pecori. 2016. S-Kademlia: A trust and reputation method to mitigate a Sybil attack in Kademlia. *Computer Networks* 94 (2016), 205–218.

[34] Alexander Poddey and Nik Scharmann. 2019. On the importance of system-view centric validation for the design and operation of a crypto-based digital economy. *arXiv preprint arXiv:1908.08675* (2019).

[35] Agostino Poggi and Michele Tomaiuolo. 2010. Extending the JADE Framework for Semantic Peer-To-Peer Service Based Applications. In *Developing Advanced Web Services through P2P Computing and Autonomous Agents: Trends and Innovations*. IGI Global, 18–35.

[36] Protocol Labs. [n.d.]. *go-libp2p*. https://github.com/libp2p/go-libp2p

[37] Martin Purvis, Mariusz Nowostawski, Stephen Cranefield, and Marcos Oliveira. 2003. Multi-agent interaction technology for peer-to-peer computing in electronic trading environments. In *International Workshop on Agents and P2P Computing*. Springer, 150–161.

[38] Lokman Rahmani and David Minarsch. 2020. *ACN Benchmark*. https://github.com/fetchai/agents-aea/blob/v0.7.0/packages/fetchai/connections/p2p_libp2p/dht/dhtpeer/benchmarks_test.go

[39] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. 2001. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. Association for Computing Machinery, 161–172.

[40] E. Rescorla. 2018. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. RFC Editor. https://tools.ietf.org/html/rfc8446

[41] Ronald L Rivest, Adi Shamir, and Leonard Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126.

[42] Atul Singh, Miguel Castro, Peter Druschel, and Antony Rowstron. 2004. Defending against Eclipse Attacks on Overlay Networks. In *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop* (Leuven, Belgium) *(EW 11)*. Association for Computing Machinery, New York, NY, USA, 21–es. https://doi.org/10.1145/1133572.1133613

[43] Sung Keun Song, SeungWok Han, and Hee Yong Youn. 2007. A new agent platform architecture supporting the agent group paradigm for multi-agent systems. In *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'07)*. IEEE, 399–402.

[44] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. 2001. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *SIGCOMM Comput. Commun. Rev.* 31, 4 (Aug. 2001), 149–160. https://doi.org/10.1145/964723.383071

[45] Antonio Tenorio-Fornés, Samer Hassan, and Juan Pavón. 2018. Open Peer-to-Peer Systems over Blockchain and IPFS: An Agent Oriented Framework. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems* (Munich, Germany) *(CryBlock'18)*. Association for Computing Machinery, New York, NY, USA, 19–24. https://doi.org/10.1145/3211933.3211937

[46] Spyros Voulgaris, Márk Jelasity, and Maarten Van Steen. 2003. A robust and scalable peer-to-peer gossiping protocol. In *International Workshop on Agents and P2P Computing*. Springer, 47–58.

[47] Peng Wang, James Tyra, Eric Chan-Tin, Tyson Malchow, Denis Foo Kune, Nicholas Hopper, and Yongdae Kim. 2008. Attacking the kad network. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks*. Association for Computing Machinery, 1–10.

[48] Shuai Wang, Liwei Ouyang, Yong Yuan, Xiaochun Ni, Xuan Han, and Fei-Yue Wang. 2019. Blockchain-enabled smart contracts: architecture, applications, and future trends. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49, 11 (2019), 2266–2277.

[49] Michael Wooldridge and Nicholas R Jennings. 1994. Agent theories, architectures, and languages: a survey. In *International Workshop on Agent Theories, Architectures, and Languages*. Springer, 1–39.

[50] Bo Zhu, Sushil Jajodia, and Mohan S Kankanhalli. 2006. Building trust in peer-to-peer systems: a review. *International Journal of Security and Networks* 1, 1-2 (2006), 103–112.