2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS) | 979-8-3503-3071-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICPADS60453.2023.00274

A Framework for Industrial Identifier Addressing Considering Compatibility and Efficiency

Yifan Xu, Fan Dang*, Jingao Xu, Xu Wang, Yunhao Liu
Tsinghua University
{xuyifan20}@mails.tsinghua.edu.cn, {dangfan, yunhao}@tsinghua.edu.cn,
xujingao13@gmail.com, wangxu.93@icloud.com

***Abstract*—Industrial Identifiers (IID), such as GS1, Handle, and OID are fundamental to device identification in growing industrial networks. Appropriate resolution and addressing methods for those identifiers are designed for wide area networks (WAN). However, due to compatibility, efficiency, and security considerations, they are not suitable for local area networks (LANs) environments. Therefore, we propose a new industrial identification framework to handle LAN scenarios by industrial address. It mainly includes the Industrial Identifier Resolution Protocol (IIRP), which combines the IIRP table lookup, the IIRP request, and the response based on the data link layer frame transmission. It is implemented as a software plug-in on LAN devices without changing any network protocol or hardware, ensuring compatibility with existing network infrastructure. Our experiments also test the efficiency of the IIRP protocol.**

***Index Terms*—Industrial Internet, Industry Identifiers, addressing**

I. Introduction

Advancements in intelligent automation are leading to radical transformations in technology, industries, and social patterns as Industry 4.0 progresses [1]. With the rapid progress of information technology (IT) and its deep integration with operational technology (OT), the Industrial Internet has emerged as a prominent research field, capturing significant attention from both academia and industry [2], [3]. The Industrial Internet empowers manufacturing and unlocks the potential of machinery, envisioning a seamless connection between the physical world and cyberspace [4].

Industrial identifiers (IIDs) play a crucial role in establishing connectivity within industrial networks. [5], [6] They serve as unique labels or addresses that allow the identification and differentiation of various components, devices, and systems in the network. They play a vital role in ensuring seamless data transmission, efficient management, and secure operation of industrial systems. One such example is the Internet Protocol (IP) address, which provides a logical address to networked devices in order to establish communication over the Internet or large networks. Besides, other IIDs have been proposed by various organizations to ensure unique identification and efficient management of products, digital objects, and resources. For instance, Global Standard 1 (GS1) provides a set of standardized identifiers for supply chain management and product identification such as Global Trade Item Numbers (GTINs), Global Location Numbers (GLNs), and Serial Shipping Container Codes (SSCCs) [7]. Other IIDs include Handle identifiers authorized by the Handle system [8], object identifiers (OIDs) standardized by the ITU and ISO/IEC [9], *etc.*

The resolution and addressing of identifiers are significant in enabling effective management and interaction of identifiable data objects, fostering seamless connectivity and data sharing within the Industrial Internet. However, traditional addressing and resolution techniques are mainly designed for Wide Area Networks (WANs), not suitable for Local Area Network (LAN) environments. There are three main reasons for this. First, these resolution methods such as Object Name Service (ONS) in GS1 and recursive queries in the Handle system rely on querying multiple layers of servers, which is inefficient and time-consuming. Second, they require accessing external servers and networks for processing, which may pose security risks such as information leakage. Third, different IID systems have their own unique resolution and addressing systems, preventing interoperability between devices from different systems.

However, according to our field study on one of the most prestigious auto glass companies worldwide (anonymity due to blind review), LAN plays a dominant role over WAN (Wide Area Network) in factories, such as Ethernet, EtherCAT, PROFINET, *etc.*. LAN offers a fast and reliable network connection within a limited area, ensuring efficient communication and real-time data exchange. Besides, LAN provides enhanced security compared to WAN. Its localized nature allows for robust security measures, protecting sensitive data and minimizing unauthorized access risks. To cope with the LAN scenario in the factories, we propose a novel IID addressing technique. It enables devices to identify the physical address of another device on the LAN in order to send and receive data based on their IIDs. The design of this system aims to tackle the following challenges. First, it should be compatible with devices of any identification type. Second, it should be compatible with existing network infrastructure.

The proposed addressing mechanism incorporates two aspects, the IID-MAC mapping and an addressing protocol design, to tackle the challenges. **1) Considering the universality of MAC addresses, we map different types of IID to MAC addresses.** As a result, we transform communication

* Corresponding Author

2690-5965/23/$31.00 ©2023 IEEE
DOI 10.1109/ICPADS60453.2023.00274

based on IIDs into communication based on MAC addresses. Devices with network interfaces, such as Ethernet cards or WLAN cards have a mapping between their IID and a certain MAC address. As for devices without MAC addresses (*e.g.,* passive tags), their relevant information and data are stored on a LAN gateway, which is common in industrial networks. In this case, the IID of the device is mapped to the MAC address of the gateway, and the mapping information is stored in the gateway as well. This ensures compatibility with devices of any IID type. **2) Then we design an Industrial Identifier Resolution Protocol (IIRP) to retrieve the corresponding MAC address with the IID of the target device inside the LAN.** By running a combination of IIRP table lookup, IIRP request, and response, we ensure timely and accurate IID-MAC mapping. The requests and responses are encapsulated into link layer frames for transmission on the LAN. It is sufficient to deploy software plugins running IIRP on devices that require addressing based on IIDs, without modifying any existing network protocols or LAN devices (endpoints or switches). Furthermore, if it is desired to boost the addressing efficiency, caching mechanisms on switches can be implemented as we shall discuss in Section VI. It helps to achieve high-performance lookup operations on switches and reduces unnecessary network communication between terminal devices.

To sum up, the contributions are as follows:

- We design a novel addressing and resolution technology for Industrial Identifiers that is compatible with devices of any IID type. we enable seamless communication between devices using IIDs by automatically resolving the IIDs of destined devices to MAC addresses with our proposed mechanism.
- The presented IIRP addressing protocol is based on link-layer transmission and is compatible with existing network infrastructure, preserving the benefits it provides.
- We implement IIRP as software plugins on hosts including a PC and a Banana Pi on a LAN. Besides, we conduct abundant evaluations exploring its performance. The results reveal that it introduces a subtle execution time of at most $3.11\,\mathrm{ms}$ to the end devices. In addition, it is resource-efficient in terms of CPU and RAM utilization. Therefore, the lightweight implementation of the mechanism is applicable to prevalent resource-limited IoT devices.

The rest of the paper is organized as follows. Section II reviews the previous related work. We introduce the thorough address resolution mechanism design in Section III and the implementation details in Section IV. Through extensive experimental results, we evaluate the performance in Section V. Finally, we leave a couple of discussions in Section VI and conclude the paper in Section VII.

II. Related Work

In Industrial Internet management, the presence of a resolution and addressing system is crucial. This section provides a comprehensive review of the pertinent technologies.

A. Resolution and addressing technology for mono-identifiers

DNS (Domain Name System) is a fundamental technology that translates domain names into IP addresses, enabling users to access resources on the internet using human-readable names [10], [11]. While DNS continues to undergo enhancements to improve its performance and capabilities, it is primarily designed for communication at the host level. However, the current Industrial Internet demands interoperability at the level of digital objects, making DNS not entirely suitable for the current IoT.

With the proliferation of industrial networks, several industrial identifier systems have been proposed, together with their corresponding resolution and addressing techniques. They can be divided into the following two categories: 1) The system is based on DNS, such as ONS in GS1, OID, and Decentralized Identifier (DID). 2) The system is independent of DNS, such as Handle, Unique Identifier (UID), Global Name Service (GNS), and Blockchain Name Service (BNS). ONS within the GS1 framework extends DNS infrastructure for resolving GS1 identifiers to relevant addresses that refer to information about products, services, or locations [12]. When resolving an OID, the identifier is parsed from left to right, traversing the hierarchy until the specific object or concept is reached [13]. DID is a self-sovereign and decentralized system [14]. Resolving a DID involves querying the appropriate network or system to retrieve the associated identity information and cryptographic keys related to the given DID. The resolving in DID, EPC, and OID is HTTP-based. Therefore the response delay is roughly the sum of the DNS domain name resolution time, the TCP connection establishment time, and the HTTP transaction time, which is relatively high. Besides, they also face the risks brought by DNS, like DDOS attacks [15].

Both the Handle and UID systems operate independently of DNS and offer identity resolution services through recursive resolution using TCP and UDP protocols [8], [16]. As distributed alternatives to DNS, GNS and BNS provide similar functionality to DNS, mapping domain names to IP addresses but without any central root servers. Even though they get rid of the DNS infrastructure, the complex resolution process such as recursive requests, and blockchain computation still hampers the addressing efficiency.

Above all, all of the mentioned resolution systems face Compatibility problems. Each of them is only able to handle a specific type of IID.

B. Resolution and addressing technology for multi-identifiers

To boost compatibility, several designs of resolution and addressing systems for multiple IIDs have been put forward.

Liu *et al.* proposes a general identification and resolution architecture for the Industrial Internet [17], and study three key technologies, namely, an identifiable digital object (IDO) model, a hybrid structure-based identification and resolution system, and a trustable system based on blockchain. Wang *et al.* proposes an IID management strategy based on

TABLE I: A possible IIRP table on a host in LAN

IID type	IID	MAC address	Adapter type	TTL
GS1	(00)006141411234567890	0FA1C2B36578	Ethernet	15:20:44
Handle	86.1009.2000/0001.1234567	AC1234F213CD	802.11	15:30:34
OID	1.2.156.3001.05.01.1001.01	ABA1C2B36578	Ethernet	15:22:44
ECODE	100036901234567892	CDA1C2B36578	802.11	15:21:44

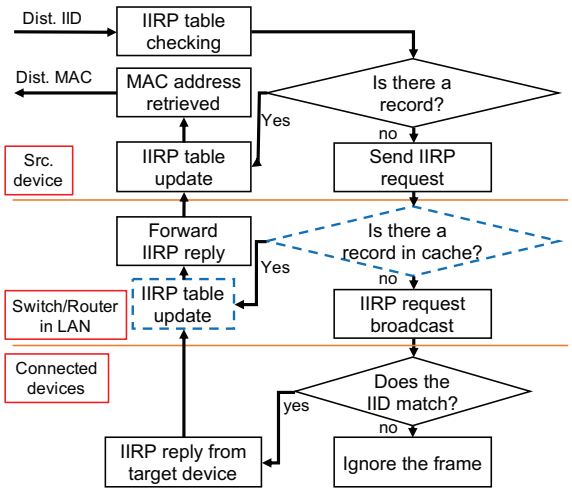

Fig. 1: Overview of IIRP protocol, which resolves any kind of IID to MAC address.

multi-identifier network architecture [18]. An inter-translation scheme between multiple identifiers based on a recursive query and an addressing and routing algorithm for identifier resolution are also designed. MIN-IIoT introduced in [19] is a scalable identifier system for industrial internet based on multi-identifier network architecture (MINA).

However, the proposed mechanisms are mainly designed for Wide Area Networks (WANs). Accessing external servers and networks may pose security risks such as information leakage for industrial networks inside factories.

III. The Design of IID Addressing Mechanism

Suppose that a host with identifier *IID_1* wants to send a datagram to another host with identifier *IID_2* on the LAN. The sender must give its adapter the MAC address for the destination and the adapter will then construct a link-layer frame containing the destination's MAC address and send the frame into the LAN. The important question that should be addressed is, how does the sending host determine the MAC address for the destination host with *IID_2*? To solve the problem, we propose the IID addressing mechanism. An addressing module in the sending host takes any IID of the destination host as input and returns its corresponding MAC address.

A. Mechanism overview

The addressing mechanism incorporates two parts, the IID-MAC mapping and the IIRP protocol to retrieve and update the mapping on devices. The universal solution is intended to be practical for any kind of LAN, whether wired or wireless.

Each host stores an IIRP table in its memory, which contains mappings of IID to MAC addresses as shown in Table I. Now suppose that host with *IID_1* would like to send a datagram to another host or router with *IID_2* on that subnet. The sending host needs to obtain the MAC address of the destination given the *IID_2*. It follows the IIRP protocol to fulfill its aim as shown in Fig. 1.

The solid black boxes in the figure describe the general address resolution process. If the sender's IIRP table contains an entry for the destination node, the target MAC address can be obtained directly. In case the IIRP table does not currently have an entry for the destination, the sender generates a specialized packet known as an IIRP request packet (described in the subsequent subsection). The purpose is to inquire about the MAC address corresponding to the resolving IID by reaching out to all the other hosts and routers within the subnet. Subsequently, the sender forwards the IIRP query packet to the adapter, indicating that the packet should be sent to the MAC broadcast address, namely, FF-FF-FF-FF-FF-FF. The adapter encapsulates the IIRP packet within a link-layer frame, utilizing the broadcast address as the frame's destination address, and transmits it into the subnet. All the other adapters within the subnet receive the frame containing the IIRP query, and each adapter relays the IIRP packet within the frame to its IIRP process module. The module verifies if its IID matches the destination IID specified in the packet. The module that finds a match sends a response IIRP packet back to the querying host, providing the desired mapping. The querying host can then update its IIRP table and send its datagram, encapsulated in a link-layer frame with the destination MAC address.

In addition, the dashed blue boxes in Fig. 1 can be an option for switches or routers in LAN to speed up the IID resolution process. They can cache an IIRP table as well, recording the IID-MAC mapping by inspecting the IIRP traffic passing by. After that, they can directly initiate an IIRP reply if there is a corresponding IID-MAC mapping entry in the cached IIRP table for an arriving IIRP request, and therefore reduce redundant IIRP traffic. However, there is a trade-off between device compatibility and addressing efficiency as we shall discuss in the following Section III-D.

B. IIRP table management

The IIRP operates in a plug-and-play manner, meaning that an IIRP table is constructed automatically without requiring configuration by a system administrator. Additionally, if a host

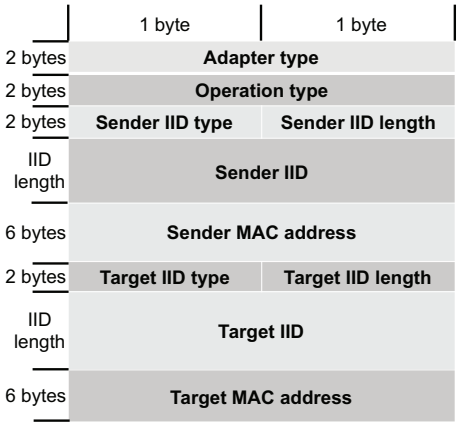

Fig. 2: Format of an IIRP packet containing one address resolution request or response, which is carried at the data link layer of the underlying network as raw payload.

Preamble	Destination MAC	Source MAC	EtherType (IIRP)	Data (IIRP packet)	CRC

Fig. 3: Format of an Ethernet frame encapsulating an IIRP packet, where a new EtherType should be applied.

becomes disconnected from the LAN, its entry is eventually removed from the IIRP tables of other devices on the LAN.

Considering that different IID systems may authorize the same IID to different devices, each entry in the designed IIRP table not only records the IID but also specifies the type of the IID as depicted in Table I. Since the MAC address may come from different kinds of network interfaces, like Ethernet or 802.11 NICs. The type of adapter is also recorded in the table so that the sender with various types of NICs can choose the proper one to communicate with the device of the destination MAC address. The IIRP table also includes a time-to-live (TTL) value, specifying the time point at which the corresponding entry will be removed from the table. It's important to note that the table may not have an entry for every host and router on the subnet. Some may have never been added to the table, while others may have expired and been subsequently removed. The number of entries in the table needs to be determined based on the device's memory size, and the duration of each entry can also be set according to specific requirements. The specific configuration of our implementation will be described in the following section.

C. IIRP packet format

As illustrated in Fig. 2, the IIRP employs a packet format that consists of a single address resolution request or response. These packets are transmitted at the data link layer of the underlying network as raw payload. Therefore, the design is independent of and applicable to diverse link layer technologies.

The adapter type denotes the network link protocol type, similar to the HTYPE field found in ARP packets [20]. The

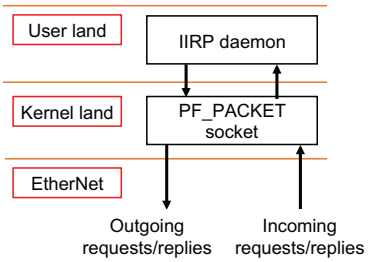

Fig. 4: The structure of IIRP

operation type specifies the sender's action, with 1 indicating a request and 2 indicating a reply. The IID type should be agreed upon by all devices, such as 0 for GS1, 1 for Handle, 2 for OID, and so on. Together with the IID length field, they guarantee compatibility with diverse lengths of different types of IIDs. In an IIRP reply, the sender MAC address field specifies the address of the host sought by the original request. Regarding the target MAC address field, it is disregarded in an IIRP request. In an IIRP reply, this field indicates the address of the host that initiated the IIRP request.

When it comes to the data link layer, it is essential to consider the MAC address and frame type. For instance, let's take Ethernet as an example, which is widely used as a link layer protocol in industrial networks. A typical Ethernet frame encapsulating an IIRP packet is depicted in Fig. 3. To distinguish the IIRP protocol from other Ethernet frames, a new *EtherType* should be registered for IIRP request and response. In an IIRP request frame, the destination MAC field should be set to the MAC broadcast address (FF-FF-FF-FF-FF-FF), while the field in the reply frames should be filled with the MAC address of the host that initiated the request, thereby making the IIRP reply a unicast message.

D. Trade-off between compatibility and addressing efficiency

As we describe in Section III-A, it can be an option for switches and routers in the LAN to additionally manage an IIRP table locally in the cache. The cache design helps speed up the resolution process and save network resources. When receiving an IIRP request, they quickly search its local IIRP table at first. If there is a corresponding record, the switch extracts the destination MAC address and returns it to the source device, so that the extra traffic searching the target device for the IIRP request in the LAN is saved. In addition, a hardware-based lookup algorithm that leverages Ternary Content-Addressable Memory (TCAM) can be implemented to further speed up the IIRP table look-up operation.

However, compatibility is sacrificed by those means, because switches or routers have to make extra programming to support the IIRP table management function, which is not yet supported by COTS (Commercial Off The Shelf) devices. Neither does the IIRP table design based on TCAM for hosts or switches/routers. If compatibility is preferred over efficiency, the protocol can be implemented just on the needed hosts as software plugins. The switches and routers in the

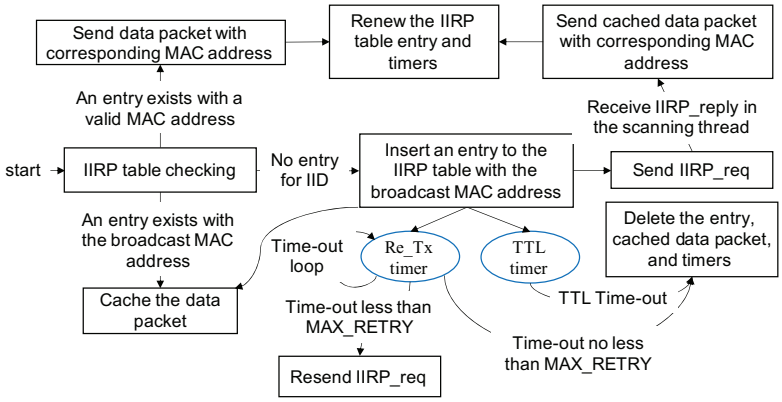

Fig. 5: The implementation of the process sending data packets to devices with IIDs

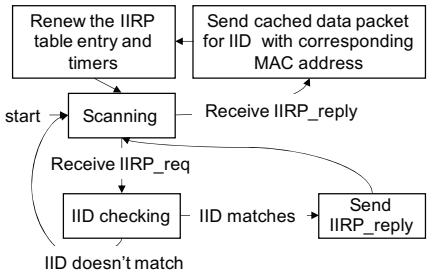

Fig. 6: The implementation of the scanning thread

LAN just play parts in data forwarding without any modification. The IIRP requests and responses are encapsulated in standard link-layer frames, and hence the COTS switches and routers are able to forward them to the right ports precisely.

IV. The Implementation of IIRP on Hosts

Based on our addressing mechanism design, we implement the IIRP protocol as software plugins on hosts. This section expands on the implementation in detail.

A. Basic setup

We have implemented the IIRP protocol plugins on the Linux operating system. As a proof of concept, we select Ethernet as the link layer protocol to encapsulate IIRP packets. Fig. 4 illustrates the interaction between the IIRP daemon and the Linux kernel. The daemon converts an IIRP request/reply into an array representing the Ethernet frame and forwards it to the Linux Networking kernel. Subsequently, the kernel takes control and transmits the frame over Ethernet via the designated adapter using a socket. Meanwhile, the IIRP daemon continuously scans another socket, actively listening for incoming IIRP requests or replies. Further implementation will be detailed in the subsequent subsection.

B. Implementation details

First and foremost, we utilize a linked list and a hash table to implement the IIRP table. We use the BKDR hash fuction [21] to transfer the key (*i.e.,* the IID type together with IID value) to a value of 0-99, indicating the index of the hash table. Each element of the hash table points to a linked list, where each linked node represents an IIRP entry with the key. The hash method ensures a fast lookup of the IIRP table. Additionally, the linked list offers improved efficiency for inserting or updating entries compared to an array.

Secondly, the protocol implementation consists of two main parts: the process of retrieving an IIRP-MAC mapping and a thread that monitors IIRP requests and replies.

Fig. 5 outlines the process of obtaining the IIRP-MAC mapping when a host attempts to send a data packet to another device using its IID. Initially, it checks the IIRP table. **1) If there is no entry for the target IID**, a new entry is inserted into the IIRP table with the MAC address left vacant, using the broadcast address. This address will be updated with an IIRP reply received subsequently. The data packet is then cached in a queue, awaiting transmission once the IID-MAC pair is retrieved. Simultaneously, an IIRP request is initiated through Ethernet using a socket. A one-time TTL timer and a periodic Re_Tx timer are started. The TTL timer counts down and times out when the entry should be removed based on the specified TTL configuration in Section III-B. The Re_Tx timer controls the retransmission of IIRP requests. If an IIRP request is sent without receiving a corresponding IIRP reply within the time interval of **REQ_TIMEOUT**, the timer times out. If the number of timeouts does not exceed the configured **MAX_RETRY**, the host resends the IIRP request. Otherwise, the entry, along with the cached data packets and timers, is released. This retransmission mechanism ensures robustness against packet loss due to network fluctuations. Upon receiving an IIRP reply in the scanning thread, the host extracts the IID and MAC address from it, sends the data packets in the cache as Ethernet frames with the MAC address, and updates the corresponding IIRP table entry and timers. **2) If an entry already exists for the desired IID**, the host examines the MAC address value in the entry. It is possible that the IIRP reply has not arrived for a previously sent IIRP request, and the MAC address in the entry remains the broadcast address. In this case, the data packet is cached, anticipating the arrival of an IIRP reply and the update of the IID-MAC mapping. If the entry contains a valid MAC address other than the broadcast address, the host can directly send the data packet in an Ethernet frame. Furthermore, the TTL in the entry, as well as the timers, should be renewed after accessing the entry.

In addition to matching MAC addresses with IIDs for data transmission, devices on the LAN should be able to respond to incoming IIRP instructions, as depicted in Fig. 6. A thread is created to constantly scan an Ethernet socket. Upon receiving an Ethernet frame containing an IIRP reply packet, the appropriate actions are taken as mentioned in the

TABLE II: Execution time of microseconds for the important steps in the IID resolution process.

Device	Step	Min(us)	Max(us)	Mean(us)	Std.dev(us)
PC	Search entry	0	3	0.88	0.53
	Insert entry	14	105	47.87	18.02
	IIRP RTT	696	2800	890.59	142.09
	Total process	712	2821	945.93	143.06
BPI	Search entry	1	5	1.98	0.62
	Insert entry	33	240	75.46	19.59
	IIRP RTT	745	2986	1268.36	173.08
	Total process	842	3111	1356.71	182.13

previous paragraph. When an IIRP request is received, the IID field in the packet is checked, and a mismatch results in the packet being ignored. Otherwise, an IIRP reply packet is sent in a unicast Ethernet frame.

It is important to note that several parameters mentioned above should be configured based on practical requirements, including the length of IIRP table, the TTL for each entry, REQ_TIMEOUT, and MAX_RETRY for the Re_Tx timer.

V. Evaluation

In order to accurately evaluate the performance of IIRP, we implement the protocol as software plugins on both a PC and an IoT device. The PC used for testing is equipped with an Intel Core i7-10710 processor, 16 GB RAM, and running the Ubuntu 20.04 operating system. It is wire-connected to a Banana Pi M2S (BPI) device [22], which features an Amlogic A311D processor, 4 GB RAM, and also runs Ubuntu 20.04.

It is important to note that our implementation is designed to function effectively in various LAN topologies. The inclusion of switches or routers between the hosts only affects the network transmission latency, which does not impact our intrinsic addressing mechanism design. Therefore, for our experimental evaluations, we have conducted tests with the hosts directly connected by wiring their Ethernet ports.

We have performed two sets of measurements. Firstly, we measured the execution time of the IID resolution process. Subsequently, we compared the time delay caused by the IIRP protocol with that of the Address Resolution Protocol (ARP) [23], which is a conventional method used to discover the link-layer address associated with a given IPv4 address.

A. Time efficiency of the IID resolution process

From a performance perspective, the execution time of the IIRP (Internet Identifier Resolution Protocol) is determined by several key steps: IIRP entry lookup, IIRP entry insertion, and the communication process for IIRP requests and replies. We conducted 1000 tests to measure the execution time of each step and the entire IID resolution process on both PC and BPI. In each test, denoted as the i_th test, we inserted i entries into the IIRP table to examine whether the execution time of table management (search or insertion) would be affected by the increasing number of entries.

The results are summarized in Table II. As shown in the table, the execution time of the IIRP request and reply

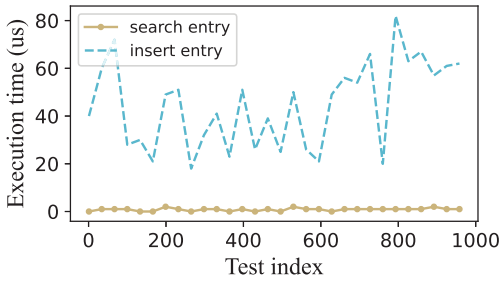

Fig. 7: The execution time for entry search and insertion during different tests on PC, with an increasing number of previously inserted IIRP entries

TABLE III: Execution time of microseconds for IIRP and ARP resolution process on PC, when there is no corresponding entry in the cache

Protocol	Min(us)	Max(us)	Mean(us)	Std.dev(us)
IIRP	712	2821	945.93	143.06
ARP	904	2624	1137.80	106.66

communication process (referred to as the IIRP RTT entry in Table II) is the dominant factor in the overall process when there is no existing IID-MAC address mapping entry in the IIRP table. The RTT varies due to network fluctuations. On average, it is measured as $890.59\,\mu s$ on PC and $1268.36\,\mu s$ on BPI. Consequently, the total execution time of the IID resolution process takes $945.93\,\mu s$ and $1356.71\,\mu s$, with only $55.34\,\mu s$ and $88.35\,\mu s$ of time overhead, respectively.

Regarding the entry lookup and insertion processes, which are also significant parts of the entire IID-MAC retrieval process, the former takes a few microseconds, while the latter occupies dozens of microseconds on average. Fig. 7 illustrates their specific values as the number of previously inserted entries in the IIRP table increases. The results show that the execution time for searching or inserting entries does not increase with the accumulation of existing entries in the IIRP table. This is due to the hash table design for entry lookup, which guarantees a computational complexity of O(1), and the head insertion method in the linked list, which ensures a constant time complexity as well for entry insertion. On the other hand, if there is an existing IID-MAC address pair in the cache, the host only needs to perform an IIRP table lookup operation for IID resolution. In this case, the total execution time is equal to the time taken for the entry search, resulting in the addressing process being completed immediately in a few microseconds.

From another perspective, the IID addressing execution on BPI takes longer than on PC due to the slower processor. However, the total runtime for IIRP, including the request and reply communication, does not exceed $3.2\,ms$. This indicates that our designed IID addressing mechanism is time-efficient and can be not only applied to powerful PC but also to

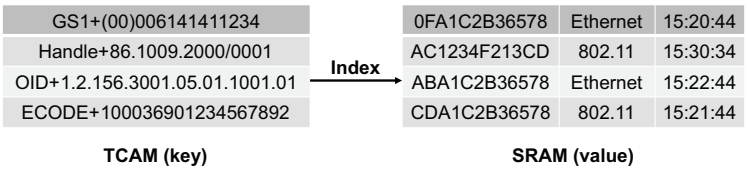

Fig. 8: An example of IIRP table caching using TCAM and SRAM

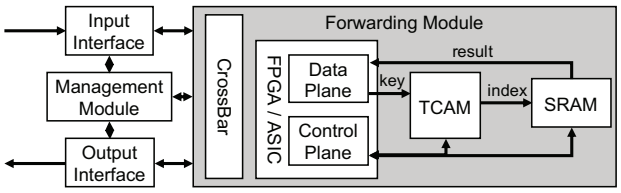

Fig. 9: A possible switch modification approach to boost IID addressing efficiency.

prevalent IoT devices with limited computation resources.

B. The comparison of execution time between IIRP and ARP

To further demonstrate the efficiency of our IID addressing mechanism, we compared the runtime of IIRP with ARP. In cases where there is an existing IID-MAC address mapping in the cache, the resolution process is very brief, involving a simple table lookup operation. To better observe the difference, we conducted the evaluation under the configuration where there is no corresponding IID-MAC or IP-MAC mapping in the cache, requiring communication within the LAN to retrieve the mapping.

Additionally, we performed 1000 ARP tests on the PC to obtain the relevant MAC address by the IP address of BPI. The comparison results are summarized in Table III. The results demonstrate that our design mechanism is even more time-efficient than ARP, with an average execution time that is 191.77 microseconds (20.27%) shorter. Considering that mainstream networked devices are capable of running the ARP protocol and can quickly resolve IP addresses to MAC addresses, running the IIRP protocol would not pose a significant burden for them.

C. Resource efficiency of the IID resolution process

Finally, we measured the CPU and RAM utilization during the IID resolution process. We conducted the process on both PC and BPI, each with 1000 pre-inserted IIRP entries. The minimum unit of CPU and RAM occupation is 0.1%. Based on our observation, the CPU usage remained below 0.1% on both devices, while the RAM usage was below 0.1% on the PC and 0.2% on the BPI.

Therefore, our proposed IID resolution process can be implemented as lightweight software plugins that are resource-friendly for IoT devices with limited resources.

VI. Discussion

A. Speed up IID addressing

As we introduce in Section III-D, an additional cache design on switches or routers helps to optimize the IID addressing process and enhance its efficiency. In this section, we first discuss the detailed practical high-performance cache lookup operations and the modification to the devices, using programmable switches as an example.

A hardware-assisted lookup algorithm utilizing TCAM by combining TCAM-based index retrieval with SRAM-based result retrieval is a good choice. Since the switches have to deal with much larger traffic than hosts, we target them as the first priority. By the way, it can be applied on the hosts or routers as well if they would like to speed up the IID resolution process.

The hardware-assisted lookup algorithm takes advantage of TCAM, a specialized type of memory. It has the ability to simultaneously compare a desired pattern against all pre-stored entries, allowing for a single clock cycle search operation across the entire entry list, making it ideal for fast table lookup [24]. For startup, the virtual IIRP table is separately stored in TCAM and SRAM. As shown in Fig. 8, each TCAM entry includes the IID (type and value) and the index pointing to an SRAM address where the corresponding MAC address, adapter type, and TTL are stored.

Fig. 9 exhibits the corresponding switch modification. A little incrementation to the forwarding module should be made, using FPGA for prototype validation and rapid development, and then transitioning to ASIC for mass production. When an IIRP request is received through the input interface and the destination IID is extracted from the frame by the Data Plane, the lookup process begins by using the IID as the input key. The TCAM performs a parallel search and returns the index of the matching entry. It is then used to retrieve the corresponding MAC address from the result in SRAM.

Self-learning and table update The switch incorporates a **self-learning** mechanism similar to the traditional switch table self-learning process to update the IIRP table including TCAM and SRAM, ensuring that it accurately reflects the current mappings between MAC addresses and IIDs. This self-learning process utilizes the traffic from hosts and other switches that pass through the switch (IIRP requests and replies). During the self-learning process, the switch updates the IIRP table by adding or updating entries of TCAM and the content in the corresponding SRAM address based on the observed MAC addresses and IIDs. Even though

the unique hardware structure of TCAM enables its greater lookup efficiency than SRAM, it causes slower updates due to the Priority Order Constraint (POC), making it a bottleneck in IIRP table update [25]. Several available approaches [25], [26] can be implemented to tackle the problem.

Security concerns Currently, we present the prototype as a proof of concept and transmit IIRP packets in plaintext. However, despite LAN providing improved security for factories compared to WAN, potential security issues still exist. For example, a malicious device could be introduced into the LAN. Since IIRP lacks authentication mechanisms to validate messages and relies on caching IID-MAC mappings, it can be compromised by falsified IIRP messages sent by the malicious device over the network. Furthermore, the plaintext nature of IIRP packets makes them susceptible to spoofing or modification during transit.

Considering that key exchange algorithms would introduce additional traffic overhead to the network, they are not ideal options. In a private network, using pre-shared keys with regularly updated nonces embedded in encrypted broadcast link layer frames could be a better solution. Regarding authentication, lightweight signature algorithms such as ECDSA [27] or EdDSA [28] are possible choices.

VII. Conclusion

With the proliferation of the Industrial Internet, industrial identifiers are playing an increasingly important role in identifying ubiquitous IoT devices and establishing connectivity within industrial networks. However, the corresponding identifier resolution approaches are designed for WANs and come with compatibility, efficiency, and security problems. Therefore, we propose a novel framework for IID addressing to cope with the LANs which play a dominant role in factories. By implementing Industrial Identifier Resolution Protocol as a software plugin on LAN devices, it efficiently resolves the IID to the MAC address of the target devices, guaranteeing compatibility with the existing network infrastructure and facilitating seamless communication within industrial networks. Our experimental results prove that the mechanism is both time-efficient and resource-efficient, and thus it is applicable to the prevalent resource-limited IoT devices.

Acknowledgments

This work is supported in part by the National Key Research Plan under grant No. 2021YFB2900100, and the NSFC under grant No. 62302259.

References

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & Information Systems Engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[2] J.-Q. Li, F. R. Yu, G. Deng, C. Luo, Z. Ming, and Q. Yan, "Industrial Internet: A Survey on the Enabling Technologies, Applications, and Challenges," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1504–1526, 2017.

[3] P. K. Malik, R. Sharma, R. Singh, A. Gehlot, S. C. Satapathy, W. S. Alnumay, D. Pelusi, U. Ghosh, and J. Nayak, "Industrial Internet of Things and Its Applications in Industry 4.0: State of the Art," *Computer Communications*, vol. 166, pp. 125–139, 2021.

[4] C. Yang, W. Shen, and X. Wang, "The Internet of Things in Manufacturing: Key Issues and Potential Applications," *IEEE Systems, Man, and Cybernetics Magazine*, vol. 4, no. 1, pp. 6–15, 2018.

[5] H. Aftab, K. Gilani *et al.*, "Analysis of Identifiers in IoT Platforms," *Digital Communications and Networks*, vol. 6, no. 3, pp. 333–340, 2020.

[6] B. Da, P. P. Esnault, S. Hu, and C. Wang, "Identity/Identifier-Enabled Networks (IDEAS) for Internet of Things (IoT)," in *Proceedings of 2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, 2018, pp. 412–415.

[7] "GS1," https://www.gs1.org/, (Accessed on July 5th, 2023).

[8] S. Sun, L. Lannom, and B. Boesch, "RFC3650: Handle System Overview," 2003.

[9] P. Sousa, A. R. Silva, and J. A. Marques, "Object Identifiers and Identity: A Naming Issue," in *Proceedings of International Workshop on Object Orientation in Operating Systems*. IEEE, 1995, pp. 127–129.

[10] P. Mockapetris, "Domain Names-Concepts and Facilities," Tech. Rep., 1987.

[11] P. V. Mockapetris, "Rfc1035: Domain Names-Implementation and Specification," 1987.

[12] M. Mealling, "Auto-ID Object Name Service (ONS) 1.0," *Auto-ID Center Working Draft*, vol. 12, 2003.

[13] "Information Technology—Open Systems Interconnection—Part 1: Object Identifier Resolution System," https://www.iso.org/standard/45247.html, (Accessed on July 5th, 2023).

[14] D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant, M. Sabadello, and J. Holt, "Decentralized Identifiers (DIDs) v1.0: Core Architecture, Data Model, and Representations," *W3C Working Draft*, vol. 8, 2020.

[15] S. Ariyapperuma and C. J. Mitchell, "Security vulnerabilities in DNS and DNSSEC," in *Proceedings of The Second International Conference on Availability, Reliability and Security (ARES'07)*. IEEE, 2007, pp. 335–342.

[16] N. Koshizuka and K. Sakamura, "Ubiquitous ID: Standards for Ubiquitous Computing and the Internet of Things," *IEEE Pervasive Computing*, vol. 9, no. 4, pp. 98–101, 2010.

[17] Y. Liu, C. Chi, Y. Zhang, and T. Tang, "Identification and Resolution for Industrial Internet: Architecture and Key Technology," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 16 780–16 794, 2022.

[18] Y. Wang, H. Li, T. Huang, X. Zhang, and Y. Bai, "Scalable Identifier System for Industrial Internet Based on Multi-Identifier Network Architecture," *IEEE Internet of Things Journal*, 2021.

[19] W. Yunmin, L. Hui, X. Kaixuan, H. Hanxu *et al.*, "Identifier Management of IndustrialInternet Based on Multi-Identifier Network Architecture," *ZTE Communications*, vol. 18, no. 1, pp. 36–43, 2020.

[20] "Address Resolution Protocol (ARP) Parameters," https://www.iana.org/assignments/arp-parameters/arp-parameters.xhtml, (Accessed on July 5th, 2023).

[21] B. W. Kernighan and D. M. Ritchie, "The C Programming Language," 2002.

[22] "Banana Pi M2s," https://wiki.banana-pi.org/Banana_Pi_BPI-M2S, (Accessed on July 5th, 2023).

[23] D. Plummer, "An Ethernet Address Resolution Protocol: or Converting Network Protocol Addresses to 48. bit Ethernet Address for Transmission on Ethernet Hardware," Tech. Rep., 1982.

[24] G. Wang and N.-f. Tzeng, "TCAM-Based Forwarding Engine with Minimum Independent Prefix Set (MIPS) for Fast Updating," in *Proceedings of 2006 IEEE International Conference on Communications*, 2006, pp. 103–109.

[25] C. Luo, C. Chen *et al.*, "BubbleTCAM: Bubble Reservation in SDN Switches for Fast TCAM Update," in *Proceedings of 2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*, 2022, pp. 1–10.

[26] B. Zhao, R. Li, J. Zhao, and T. Wolf, "Efficient and consistent tcam updates," in *Proceedings of IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1241–1250.

[27] D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International journal of information security*, vol. 1, pp. 36–63, 2001.

[28] S. Josefsson and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)," Tech. Rep., 2017.