TRANSFORMING WIRELESS SYSTEMS WITH LARGE AI MODELS: TOWARD INTELLIGENT AND ADAPTIVE CONNECTIVITY

SICONG LIAO

PhD

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University
Department of Computing

Transforming Wireless Systems with Large AI Models: Toward Intelligent and Adaptive Connectivity

Sicong Liao

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
March 2025

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: Sicong Liao

Abstract

The rapid evolution of wireless communication technologies demands intelligent and adaptive solutions to address the growing complexity of cross-technology interoperability and large-scale connectivity management. Large AI models, with their ability to process vast amounts of data and learn intricate patterns, offer a promising approach to two key challenges: seamless cross-technology interoperability and reliable connectivity management in mobile IoT networks. The primary aim of this research is to harness these capabilities to create frameworks that enhance the efficiency and scalability of wireless systems. This dissertation presents two fundamental contributions: one for the cross-technology communication and the other for the network connectivity optimization. Previous reverse engineering approaches in cross-technology communication include non-reversibility, symbol misalignment, scalability issues, and over-reliance on empirical methods, making them labor-intensive and less effective for modern applications. To address these issues, we propose XiTuXi, which enables seamless cross-technology communication between WiFi and diverse IoT protocols (e.g., ZigBee, Bluetooth, LoRa) via neural machine translation. By employing Transformer models and genetic algorithm-driven forward engineering, XiTuXi automates the translation of bit sequences across heterogeneous protocols, eliminating reliance on manual reverse engineering. In the context of network connectivity optimization, we concentrate on reducing disconnection risks within large-scale mobile IoT networks that utilize eSIM technology. To achieve this, we introduce NeSIM, which employs a lookahead multi-operator access strategy to predict operator switch events in advance,

thereby optimizing proactive profile provisioning and minimizing handover latency.

Publications Arising from the Thesis

1. Sicong Liao, Zhenlin An, Qingrui Pan, Xiaopeng Zhao, Jingyu Tong, and Lei Yang, "XiTuXi: Sealing the Gaps in Cross-Technology Communication by Neural Machine Translation", in Proc. of *ACM SenSys*, 2023.

2. Jingyu Tong, zhenlin An, Xiaopeng Zhao, Sicong Liao, and Lei Yang, "In-Sensor Machine Learning: Radio Frequency Neural Networks for Wireless Sensing", in Proc. of *ACM MobiHoc*, 2024.

2. Sicong Liao, Fei Yuan, Lei Yang, and Xuan Song, "Scaling Mobile IoT with eSIM: A Four-Year, Nationwide Study on a Ten-Million Scale", in *ACM SIGCOMM*, 2025 (Under Review).

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Lei Yang, for his unwavering support, guidance, and encouragement throughout this research journey. His profound expertise, insightful feedback, and patience have been instrumental in shaping this thesis. I am truly fortunate to have had the opportunity to work under his mentorship, which has not only helped me grow as a researcher but also inspired me to strive for excellence in every aspect of my work. His dedication to fostering a collaborative and intellectually stimulating environment has made this journey both rewarding and transformative.

I am also immensely grateful to my colleagues and collaborators in the TagSys Group for their invaluable contributions and camaraderie. Their willingness to share knowledge and engage in stimulating discussions has greatly enriched this work. The collaborative spirit and shared commitment to excellence within the team have made this research not only productive but also enjoyable.

I would also like to extend my heartfelt thanks to my family for their constant encouragement and understanding throughout this journey. To my parents, thank you for instilling in me the values of perseverance and curiosity, and for always believing in my potential even when the path seemed uncertain. Your support has been a cornerstone of my resilience and determination.

Finally, I would like to acknowledge the broader research community and the authors of the countless papers and resources that have informed and inspired this work. The

collective knowledge and advancements in the fields of wireless communication and artificial intelligence have been the foundation upon which this research was built. I am deeply grateful for the opportunity to contribute to this ever-evolving field and look forward to continuing this journey of discovery and innovation.

This dissertation is a testament to the support, collaboration, and inspiration I have received from so many individuals and organizations. To all of you, I extend my sincerest gratitude.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid evolution of wireless communication technologies has significantly transformed the way we interact with the digital world. From the proliferation of Internet of Things (IoT) devices to the advent of 5G and beyond, the demand for seamless, efficient, and intelligent connectivity has never been greater. However, the increasing complexity and heterogeneity of wireless systems pose significant challenges, particularly in achieving cross-technology communication (CTC) and optimizing network performance in large-scale IoT deployments.

Traditional approaches to wireless communication often rely on manual configuration and rule-based optimization, which are not only time-consuming but also fail to adapt to the dynamic and diverse nature of modern wireless environments. For instance, in CTC, where devices using different wireless protocols (e.g., WiFi, ZigBee, Bluetooth) need to communicate directly, conventional methods such as reverse engineering are limited by their scalability, reversibility, and reliance on empirical data. Similarly, in large-scale IoT networks, the management of embedded SIM (eSIM) technology and the optimization of multi-operator access (MOA) strategies are critical for ensuring reliable and efficient connectivity, yet current failover-based approaches often result in significant latency and service disruptions.

The integration of large AI models into wireless systems offers a promising solution to these challenges. By leveraging the power of neural machine translation (NMT) and advanced time-series forecasting, AI-driven approaches can automate and optimize complex communication tasks, enabling intelligent and adaptive connectivity. For example, in CTC, AI models can learn the underlying translation rationale between different wireless protocols, allowing for seamless communication without human intervention. In the context of eSIM-enabled IoT networks, predictive analytics powered by large AI models can forecast and schedule operator switches in advance, minimizing service disruptions and reducing handover latency.

The application of AI in wireless systems is not limited to these areas. From spectrum management and interference mitigation to network slicing and resource allocation, AI has the potential to revolutionize every aspect of wireless communication. By transforming wireless systems into intelligent and adaptive networks, AI can enhance performance, scalability, and reliability, paving the way for the next generation of wireless technologies.

1.1 Background

1.1.1 Cross-Technology Communication

Cross-Technology Communication (CTC) enables direct interaction between heterogeneous wireless protocols (e.g., WiFi, ZigBee, Bluetooth) by manipulating physical-layer waveforms to bridge incompatible standards. This technique allows a WiFi transmitter to emulate target IoT signals (like ZigBee's OQPSK modulation) through precise payload engineering, enabling devices with distinct PHY/MAC layers to interoperate without hardware modifications. While traditional reverse engineering approaches achieve CTC through manual symbol-by-symbol mapping, such methods face fundamental limitations including non-reversible transformations and alignment

constraints when adapting high-speed WiFi symbols to low-speed IoT symbols.

1.1.2 eSIM

Embedded SIM (eSIM) technology revolutionizes cellular connectivity by replacing physical SIM cards with programmable chipsets embedded in IoT devices, enabling remote provisioning and dynamic operator switching through cloud-based profile management. This ecosystem comprises three core components - the eUICC (embedded Universal Integrated Circuit Card) hardware storing cryptographic credentials, the SM-DP+ (Subscription Manager Data Preparation) server for profile distribution, and the eIM (eSIM IoT Remote Manager) orchestrating over-the-air updates. Unlike traditional SIMs requiring physical replacement, eSIMs employ dual-profile architectures (universal profile for emergency roaming and business profile for local connectivity) to maintain seamless service continuity, particularly critical for mobile IoT applications like vehicle telematics and asset tracking where devices frequently cross operator boundaries.

1.1.3 WiFi

WiFi has emerged as the *de facto* wireless local area network (WLAN) standard, enabling global internet access for tens of billions of devices through the IEEE 802.11 protocol. By leveraging radio waves to extend Ethernet connectivity wirelessly, WiFi provides seamless high-speed internet access across ubiquitous consumer electronics, including laptops, smartphones, tablets, and smart appliances (e.g., televisions, security cameras, voice assistants). Its adoption is further driven by the use of unlicensed industrial, scientific, and medical (ISM) radio bands, which eliminate licensing fees and infrastructure costs associated with cellular networks, making WiFi a cost-effective solution for both personal and commercial deployments.

The decentralized nature of WiFi infrastructure—comprising access points (APs), routers, and public hotspots—ensures pervasive coverage across residential, enterprise, and outdoor environments, including transportation systems. However, WiFi' s design philosophy prioritizes high throughput over energy efficiency, a trade-off rooted in its origins as a wireless Ethernet replacement. Conventional WiFi hardware employs wide channel bandwidths (20 MHz) and computationally intensive signal processing, resulting in greater complexity and power consumption compared to low-power wireless alternatives like Bluetooth or ZigBee. This inherent energy inefficiency poses challenges for battery-dependent IoT applications, where simpler radio architectures dominate.

1.1.4 Bluetooth

Bluetooth, originally developed by Ericsson, has become the dominant standard for personal area networks (PANs), prioritizing ultra-low power consumption and short-range communication. Designed for scenarios where extended battery life is critical, it powers ubiquitous applications such as wireless headphones, smartwatches, wearables, location beacons, and portable speakers.

Bluetooth achieves its energy efficiency through streamlined radio hardware and frequency-shift keying (FSK) modulation, a digital frequency modulation technique that enables simple circuit designs. This approach minimizes both chip footprint and power requirements, making Bluetooth ideal for compact, battery-dependent devices.

Like WiFi, Bluetooth operates in the unlicensed ISM band, ensuring cost-effective deployment for end-users. However, its core design emphasis on power conservation results in significantly narrower channel bandwidths (1–2 MHz per channel) compared to WiFi. The reduced signal complexity lowers processing demands, further enhancing energy efficiency but limiting data throughput—a trade-off that positions Bluetooth as a specialized solution for low-bandwidth, intermittent communication tasks.

1.1.5 ZigBee

ZigBee is a low-power, low-data-rate wireless communication technology designed for applications that require long battery life and reliable operation in harsh environments. It operates in ISM band, similar to WiFi and Bluetooth, allowing users to leverage the unlicensed spectrum for cost-effective deployments. However, unlike WiFi, ZigBee is optimized for low power consumption rather than high throughput. This makes it ideal for battery-operated devices that need to operate for extended periods without frequent recharging or battery replacement.

The design of ZigBee emphasizes simplicity and energy efficiency. It uses narrowband channels, typically 2 MHz wide, and employs relatively simple modulation schemes such as binary phase-shift keying (BPSK) or offset quadrature phase-shift keying (O-QPSK). These techniques reduce the complexity of the radio hardware, enabling ZigBee devices to consume significantly less power than WiFi devices. Additionally, ZigBee supports mesh networking, allowing devices to relay data through multiple hops, which extends the range and improves reliability in large-scale deployments.

ZigBee devices are often used in scenarios where low data rates are sufficient, such as transmitting sensor data or control commands. Its low power consumption, combined with its ability to support large networks of devices, makes ZigBee a popular choice for IoT applications.

1.1.6 Artificial Intelligence of Things

The Artificial Intelligence of Things (AIoT) combines Artificial Intelligence (AI) with the Internet of Things (IoT) to create smarter, autonomous systems. By integrating AI technologies like machine learning and computer vision into IoT devices, AIoT enables real-time data analysis, intelligent decision-making, and adaptive behavior. This fusion is transforming industries, from smart cities and healthcare to industrial

automation, by improving efficiency, reducing costs, and enhancing user experiences. A key application of AI in AIoT is AI for wireless, where machine learning algorithms optimize wireless communication protocols, improve network performance, and enhance spectrum efficiency. For example, AI can dynamically allocate resources in WiFi or 5G networks, predict interference, and adapt to changing network conditions, ensuring reliable and high-speed connectivity.

A key advantage of AIoT is edge computing, where AI algorithms run directly on IoT devices rather than relying on the cloud. This reduces latency, minimizes bandwidth usage, and ensures faster responses, which is critical for applications like autonomous vehicles or industrial control systems. In wireless communication, edge AI enables real-time processing of data from sensors and devices, improving responsiveness and reliability. Edge AI also enhances privacy and security by keeping sensitive data on the device. However, deploying AI on resource-constrained IoT devices requires optimization techniques like tiny machine learning (TinyML) to balance performance and energy efficiency, especially in wireless applications where power consumption is a critical factor.

The future of AIoT lies in creating self-learning ecosystems, where devices continuously improve through data and user interactions. As AI and IoT technologies evolve, AIoT will drive innovation, enabling smarter, more connected environments and addressing complex global challenges. In wireless communication, advancements in AI for wireless will further enhance network intelligence, enabling adaptive, self-optimizing systems that can handle increasing demands for connectivity and data. This integration of AI and wireless technologies will pave the way for next-generation networks, such as 6G, which are expected to rely heavily on AI for seamless and efficient operation.

1.2 Contribution of the Dissertation

The dissertation' s contributions are centered around two innovative frameworks, and NeSIM, which leverage large AI models to address critical challenges in wireless communication. XiTuXi solves protocol-level interoperability via neural translation, while NeSIM optimizes network-level connectivity through predictive analytics. Together, they form a holistic framework for intelligent and adaptive wireless networks. These contributions demonstrate the transformative potential of AI in enabling intelligent and adaptive connectivity across diverse wireless environments. The ensuing contributions are presented below.

1.2.1 XiTuXi: AI-Driven Cross-Technology Communication

XiTuXi represents a groundbreaking advancement in Cross-Technology Communication (CTC) by leveraging neural machine translation (NMT) models, specifically the Transformer architecture. Traditional CTC methods rely on reverse engineering, which suffers from limitations such as non-reversibility, symbol misalignment, and scalability issues. XiTuXi addresses these challenges by automating the translation of bit sequences between heterogeneous wireless protocols, enabling seamless communication across 30 protocol combinations. This approach eliminates the need for manual intervention and significantly reduces the complexity of CTC implementation.

Beyond its technical innovation, XiTuXi demonstrates the broader applicability of AI in wireless communication. By treating CTC as a language translation problem, XiTuXi achieves high accuracy and efficiency in real-world scenarios. Its ability to facilitate communication between devices like WiFi and IoT opens new possibilities for interoperability in diverse wireless environments. This contribution not only advances the state-of-the-art in CTC but also sets a precedent for the integration of large AI models in wireless systems.

1.2.2 NeSIM: AI-Optimized eSIM Connectivity

NeSIM introduces a transformative framework for eSIM-enabled Mobile IoT by employing large AI models for predictive analytics. The current failover-based multi-operator access (FOMOA) strategy is inefficient, leading to significant handover latency and service disruptions. NeSIM addresses this by implementing a lookahead-based multi-operator access (LAMOA) strategy, which proactively schedules operator switches based on predictive insights. This approach minimizes service disruptions and ensures seamless connectivity for mobile IoT devices.

The innovation of NeSIM lies in its ability to forecast network conditions and optimize operator selection in real time. By leveraging predictive analytics, NeSIM reduces handover latency and enhances the reliability of eSIM-enabled devices. This contribution not only improves the performance of mobile IoT networks but also highlights the potential of AI-driven solutions in wireless communication. NeSIM serves as a model for future advancements in intelligent and adaptive connectivity, paving the way for more efficient and resilient wireless systems.

1.3 Organization of the Dissertation

This dissertation is organized into four chapters addressing AI-driven innovations in wireless systems. Chapter 1 introduces the challenges of modern wireless communication and outlines two core contributions: XiTuXi, a neural machine translation framework for cross-technology communication (CTC) between heterogeneous protocols (e.g., WiFi/ZigBee), and NeSIM, an eSIM-based predictive optimization system for IoT connectivity. Chapter 2 details XiTuXi ' s genetic algorithm-driven training and Transformer-based protocol translation, demonstrating its superiority over reverse-engineering methods. Chapter 3 presents NeSIM ' s large time-series model for proactive eSIM profile provisioning, validated through a nationwide IoT deployment

study. Finally, Chapter 4 synthesizes lessons on AI integration in wireless systems, proposing future directions like federated learning and edge AI, while emphasizing the transformative potential of context-aware AI models in enabling adaptive and scalable connectivity.

Chapter 2

Enabling Cross-Technology Communication via Neural Machine Translation

WiFi has become the de facto standard for billions of wireless devices connecting to the Internet. To keep up with the ever-increasing demand for higher speed and higher capacity, WiFi devices must be equipped with an expensive DSP and considerably power-consuming circuitry for processing complicated waveforms. Meanwhile, the proliferation of Internet-of-Things (IoT) applications has led to the fast growth of IoT devices in recent decades. With the stringent board space and overstretched power budget, IoT devices have to use more power-efficient, low-cost, and pint-sized wireless chips (like Bluetooth or ZigBee) instead of the WiFi standard. As all these diverse wireless technologies share the unlicensed spectrum (e.g., ISM bands), their coexistence used to be considered harmful due to mutual interference. However, recent research demonstrated the success of waveform-emulated cross-technology communication (CTC), which allows physical-level direct communication among heterogeneous wireless devices [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], such as WiFi→ZigBee (WEBee [1]),

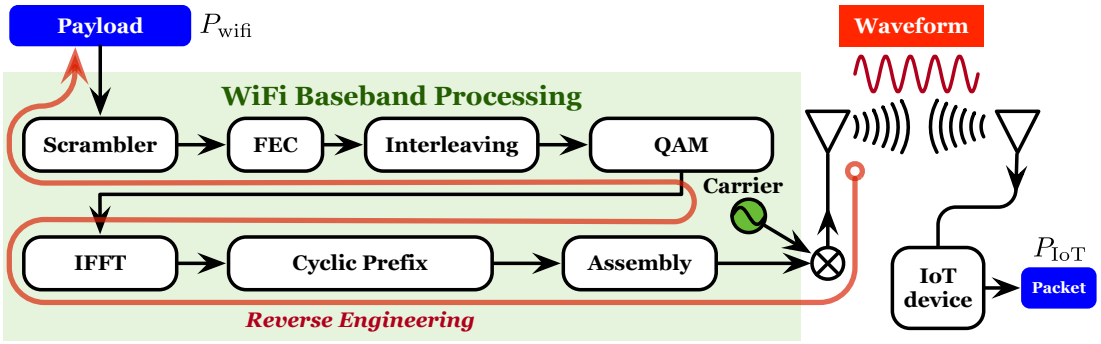


Figure 2.1: Waveform-emulated CTC from a WiFi sender to an IoT device. Past systems use reverse engineering to find an appropriate WiFi payload that can emulate the waveform of the desired IoT packet.

WiFi→Bluetooth (BlueFi [5]). These previous systems work in a simple principle: *as long as a signal's waveforms generated by a WiFi chip are close enough to those generated by an IoT transmitter (e.g., Bluetooth), the corresponding IoT receivers will be able to decode the signals correctly.* The waveform-emulated CTC shows many attractive benefits across several dimensions, including higher bandwidth utilization, higher data rate, and higher network capacity.

At the heart of waveform-emulated CTC is finding the WiFi payload, which can emulate the waveform of a desired IoT packet. Past CTC systems adopt the technique of reverse engineering to achieve this goal. To transmit the packet P_{iot} to an IoT receiver via a WiFi sender, reverse engineering reversely finds the appropriate payload P_{wifi}, which can generate a waveform similar to P_{iot}. The workflow is shown in Fig. 2.1. Specifically, $P_{\text{iot}} \rightarrow$ waveform $\rightarrow$ removal of cyclic prefix (CP) $\rightarrow$ alignment of QAM points $\rightarrow$ anti-interleaving $\rightarrow$ anti-FEC $\rightarrow$ anti-scrambling $\rightarrow P_{\text{wifi}}$. Reverse engineering-based CTC solutions have been successful.

However, reverse engineering suffers from many limitations, such as being non-reversible and unscalable, misaligning symbols, and over-relying on empiricism. For example, WiFi adopts conventional codes as forward error correction (FEC), which can be viewed as a lossy compressor. When reverse engineering attempts to recover bits fed into FEC from decoded bits, some of which will be different from the original ones

because the information is lost. This is an ill-posed problem (see §2.1 for details). These challenges force us to try all possible settings exhaustedly and even guess the input bits based on the expert's debugging experience. Although a large body of CTC works has been carried out in recent years, each of them only aims to challenge a single protocol combination (e.g., WiFi→ZigBee [1] or WiFi→Bluetooth [5]). A one-size-fits-all solution is still anticipated to librate experts from the tedious tasks.

We notice a similar cross-linguistic communication phenomenon attributed to the waveform similarity in the natural world, which is termed *homophony* – a linguistic phenomenon whereby words of different human languages become identical in pronunciation. For example, "三克油" in Chinese pronounces enormously as "thank you" in English. Actually, homophony is considerably used in translating foreign product brands, where some words are combined to form an "illogical" phrase whose pronunciation is exceptionally similar to the original Japanese, English, or French names. More examples are given as follows:

"ユニクロ"(Japanese) — "UNIQLO" (English)
"Coca-Cola" (English) — "du Coca" (French)
"Chanel" (French) — "香奈儿" (Chinese)
"Marvel" (English) — "Merveille" (French)

The above phrases are meaningless in ordinary speech but sound highly like original phrases in the corresponding other languages due to the high similarity of their acoustic waveforms. In practice, homophony-based translation benefits the quick spreading of a product's brand because the pronunciation is consistently recognized by consumers worldwide.

Inspired by the task comparability between CTC and cross-linguistic homophony, we propose a general-purpose solution called XiTuXi [1] for achieving waveform-emulated CTC from WiFi to general IoT devices, by using the mature neural machine trans-

[1] XiTuXi is the Chinese phrase pronounced similar to CTC.

lation (NMT). In this work, we reduce the CTC into a classical language translation problem, which has been studied for decades in neural language processing and has achieved remarked breakthroughs recently. The solution can also be extended for the CTC between any arbitrary protocols operating at the same spectrum. XiTuXi considers the communication devices and channel as a whole black box, so the solution can run for any 802.11-series chips instead of specific chips from particular manufacturers or for specific protocol versions. Achieving XiTuXi faces two main technical challenges:

• First, the fundamental challenge is in the acquisition of the training dataset (i.e., corpora). The great success of recent NMT benefits from the fast advance of deep learning as well as the high-volume and high-quality language corpus for the training. Similarly, XiTuXi depends on the well-annotated training dataset. Unfortunately, we face a dilemma in acquiring the training dataset. On the one hand, the CTC from WiFi to IoT devices is unidirectional, so a ZigBee packet cannot be recognized by a WiFi receiver. This condition means experiments cannot be used to acquire the samples. On the other hand, reverse engineering can help acquire the dataset. However, if reverse engineering works, the learned neural model is worthless and the challenges cannot be resolved. To address this dilemma, we introduce genetic algorithm (GA)-based *forward engineering*, which randomly generates a WiFi payload (i.e., P_{wifi}) as a chromosome and forces the target IoT receiver to decode it. If the decoded result (i.e., P_{iot}) has a highly similar waveform as the received one, it is kept for the next evolution. This process is iterated until a sufficient number of pairs of $(P_{\text{wifi}}, P_{\text{iot}})$ are acquired.

• Second, reverse engineering adopts the symbol-by-symbol replacement strategy, which requires the duration of the low-speed IoT symbol to be exactly an integral multiple of that of the high-speed symbol to meet the alignment demand. Unfortunately, the alignment is a coincidence. The reality is that a portion of a WiFi symbol may go across two IoT symbols. This condition requires us to choose appropriate

Figure 2.2: Transmitting a JEPG image via a WiFi sender using different CTC solutions. The first row shows the original picture and the pictures received by the BLE receiver; the second row shows the pictures received by the ZigBee receiver; the third row shows the received pictures when the WiFi adopts the short CP ($3.6\mu s$).

WiFi symbols based on the context, as the current NMT does. To address this issue, we adopt the well-known sequence-to-sequence NMT model called Transformer for the translation, which searches for the next WiFi symbol by taking advantage of self-attention and context-sensitive mechanisms.

Contribution. Fig. 2.2 shows the results of a case study, which compares the quality of pictures transmitted via XiTuXi and previous solutions (see §2.3.5). To the best of our knowledge, this is the first one-size-fits-all solution that takes advantage of the

recent advance of NLP to resolve the CTC problem. We demonstrate the feasibility and efficiency of XiTuXi by achieving the CTC from 11 types of WiFi protocols (i.e., versions) to 4 types of IoT protocols (i.e., ZigBee, BLE, LoRa, Sigfox). We achieve the CTC among a total of 30 combinations. All previous waveform-emulated CTC are re-explored by using XiTuXi. The results show that XiTuXi outperforms previous solutions by about twice in terms of the symbol error rate (SER) and frame reception rate (FRR).

2.1 Primers

In this section, we introduce the background and summarize the limitations of current solutions.

2.1.1 Background of WiFi

We start by introducing the background of the WiFi technique. Fig. 2.1 shows the block diagram of a WiFi transmitter in accordance with 802.11 series protocols. The incoming bit stream transmitted from the MAC layer is called the MPDU, which will be encapsulated into the payload field as PSDU in the physical layer. **(1) Scrambler**. The MPDU is fed into a scrambler to avoid the presence of a long sequence of identical bits. **(2) FEC**. FEC allows the restoration of a transmitted bit stream by inserting redundancy into the scrambled stream, even if it has experienced minor corruption due to channel interference. **(3) Interleaving**. Interleaving is used to scatter the bursts of errors by separating bits in a small vicinity to larger distances and vice versa. **(4) Modulation (QAM)**. A WiFi channel is assigned with a 20 MHz spectrum, which is divided into 64 subcarriers. Each subcarrier occupies about $20/64 = 312.5$ KHz. Taking 802.11n as an example, 52 out of 63 subcarriers are used for data transmission. **(5) IFFT**. Afterward, the IFFT converts the samples on the

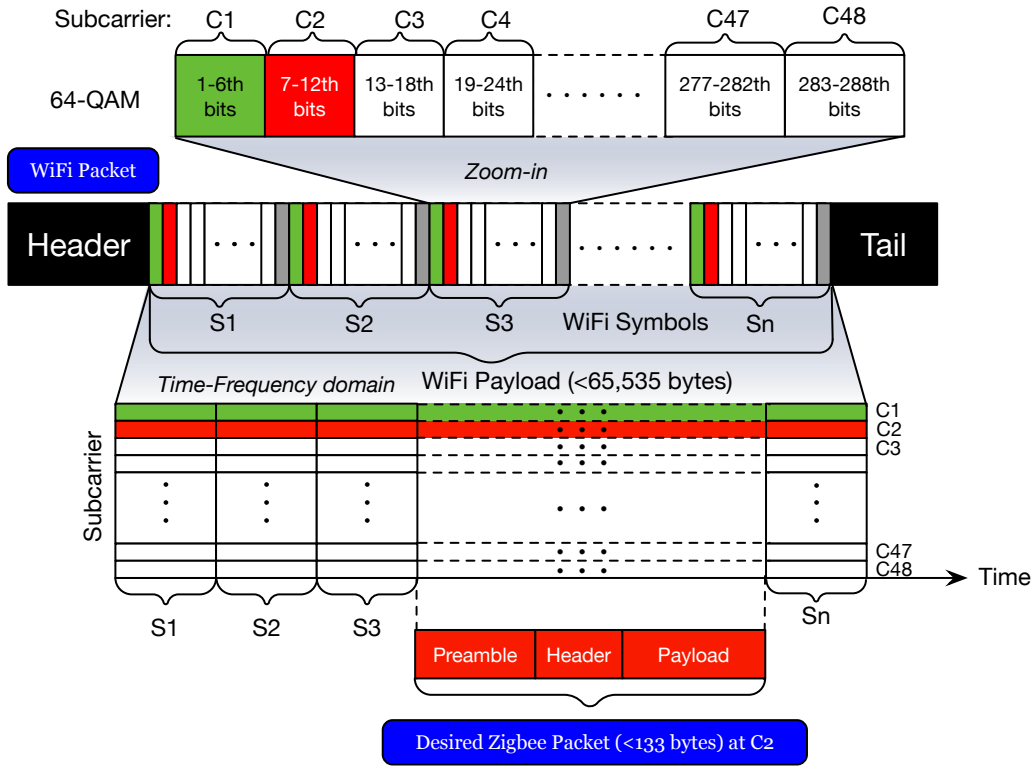


Figure 2.3: Structure of a WiFi packet in PHY. A WiFi symbol piggybacks 288 bits, among which every six continuous bits are mapped onto a subcarrier. Suppose a ZigBee receiver listens at C2, the CTC solution inserts the desired bits onto the subcarrier. As long as the final waveform on C2 is close enough to that of the desired ZigBee packet, the ZigBee receiver can recognize it.

64 subcarriers in the frequency domain to a 64-sample-long signal in the time domain. **(7) Cyclic prefixing (CP)**. The time-domain signal is then processed for the cyclic prefixing, where the last 16 samples are copied and inserted into the beginning of the 64 samples. The total $64 + 16 = 80$ samples constitute an OFDM symbol. **(6) Assembly**. After all bits from an MPDU are processed, they are placed into the payload field (i.e., PSDU) with the frame packet header and tail to assemble an entire WiFi packet. **(7) Propagation**. Finally, a whole WiFi packet is propagated into the air by the wireless frontend.

2.1.2 Waveform-Emulated CTC

IoT devices prefer simple modulation schemes (i.e., OQPSK for ZigBee; GFSK for Bluetooth) for saving energy, which can be viewed as simplified cases of OFDM. This condition allows us to use the advanced OFDM waveforms to emulate the relatively primitive waveforms generated by IoT devices. As shown in Fig. 2.3, suppose the current ZigBee receiver listens at the WiFi Subcarrier 2, we can change the 7-12th bits in each WiFi symbol such that the bits on this subcarrier (highlighted in red) are precisely assembled as an entire packet recognized by the ZigBee receiver. IoT devices can correctly decode the emulated signals once these signals are close enough to the desired waveforms. Thus, the fundamental problem is finding a WiFi payload whose final waveform at target subcarriers can emulate the desired waveforms. The problem is formalized as follows:

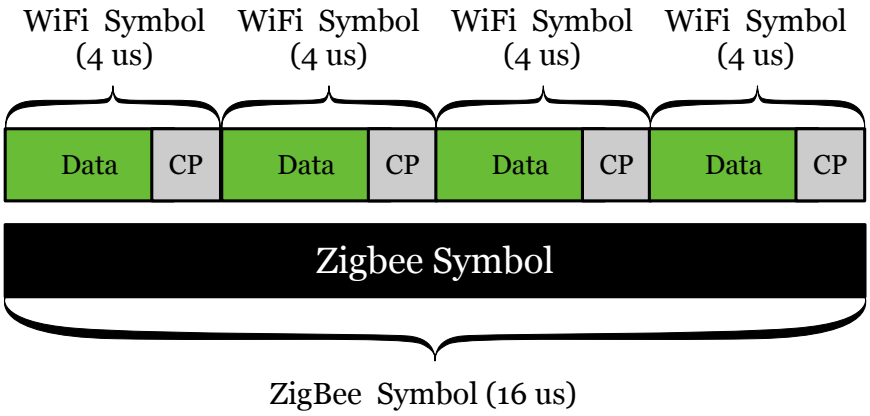
Problem 1. *Given the bits P_{iot} desired to be received at an IoT receiver, how to find bits P_{wifi} transmitted from a WiFi sender, making OFDM(P_{wifi}) $\approx$ IoT(P_{iot})?*

where OFDM($\cdot$) and IoT($\cdot$) represent the modulation schemes for the WiFi and the IoT devices, respectively. The approximation means the output waveforms are highly similar.

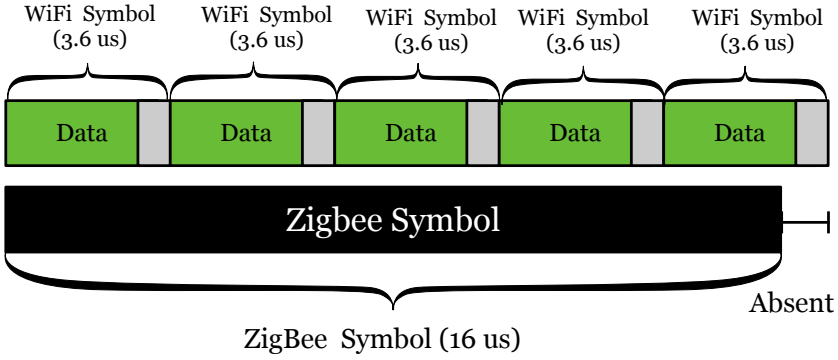
2.1.3 Limitations of Reverse Engineering

To resolve the above problem, previous works use the technique of reverse engineering. Unfortunately, it encounters many practical limitations as follows.

Limitation I: Non-reversible. Some modules in the WiFi baseband are actually non-reversible. For example, WiFi uses convolutional coding as its FEC coder to deal with random noise. The encoding process can be modeled as the following equation in Galois field GF(2): $M \times_{\mathrm{GF}(2)} X = Y$ where X and Y are the source and coded bits. The FEC coder introduces the redundancy to improve the robustness, so the



(a) Alignment



(b) Misalignment

Figure 2.4: Symbol-by-symbol emulation. (a) Four $4\mu s$ WiFi symbols with long CPs can exactly align with a single $16\mu s$ ZigBee symbol. (b) Four WiFi symbols with short CPs cannot align well with a WiFi symbol.

dimension $|Y| > |X|$, e.g., $|X| = 216$ and $|Y| = 288$ in the setting of 64-QAM and 3/4-coding rate. Notedly, the matrix M (i.e., $|M| = 218 \times 216$) is non-reversible because it is not full row-rank (row: 288 > column: 216) [1]. To deal with this issue, previous works (e.g., WEBee [1] or BlueFi [5]) have to alter bits in both X and Y, making the above equation hold approximately. Thus, the naive reversion does not work.

Limitation II: Symbol misaligned. Reverse engineering adopts the symbol-by-symbol replacement strategy without understanding the context. It usually combines an integral multiple of high-speed WiFi symbols to emulate a low-speed IoT symbol. For example, in WEBee [1], each WiFi symbol takes $3.2\mu s$ (data)$+0.8\mu s$ (CP) $= 4\mu s$,

whereas each ZigBee symbol at 250 kbps rate (the maximal rate) takes $4/250 = 16\mu s$. We can use $16/4 = 4$ WiFi symbols to emulate a single ZigBee symbol, as shown in Fig. 2.4(a). Unfortunately, the ideal alignment is a coincidence. Actually, the WiFi symbol may be set to $3.2\mu s$ (data) + $0.4\mu s$ (CP) = $3.6\mu s$ with a short CP, and the rate-adaptive IoT devices may adopt a lower data rate (e.g., 40 kbps). In these settings, the duration of a ZigBee symbol is not an integral multiple of WiFi symbols. Fig. 2.4(b) shows the case where the WiFi adopts the $0.4\mu s$ CP. Particularly, WiFi 6 (802.11 ax) offers more options on the symbol duration (i.e., $0.8\mu s$, $1.6\mu s$ and $3.2\mu s$), in which it is impossible to make two types of symbols aligned in the time.

Limitation III: Unscalable. As first introduced in 1997, the WiFi standard evolved from an 802.11b (i.e., WiFi G1) to 802.11 ax (WiFi G6) in the past 25 years. The maximum link rate increases from 11 Mbit/s to 1 G Mbit/s. The physical-layer modulation mechanism crosses CCK, FHSS, and OFDM. These six types of WiFi devices still co-exist on the market nowadays. As a result, there are more than 30 protocol combinations for direct communication from WiFi to IoT. So far, only seven combinations have been achieved only. Achieving all possible CTC combinations with a bare-handed approach makes us overburdened. An automatic solution is demanded.

Limitation IV: Over-relying on empiricism. At present, successful reverse engineering heavily relies on an expert's experience and proficiency skills. For example, the authors made two important contributions as claimed in BlueFi [5]. The first is an empirical formula used to compensate for the cyclic prefix. The second is to assign empirical weights to the coded bits when reversely finding the source bits in the FEC module. All these empirical settings must be refined from long-term engineering experience in diagnosing, debugging, and troubleshooting. We demand a prompt solution to liberate engineers from the painful task of reverse engineering.

In summary, previous works based on reverse engineering demonstrated the feasibility of CTC and its success. However, this approach suffers from the unscalable problem and inefficiency in handwork.

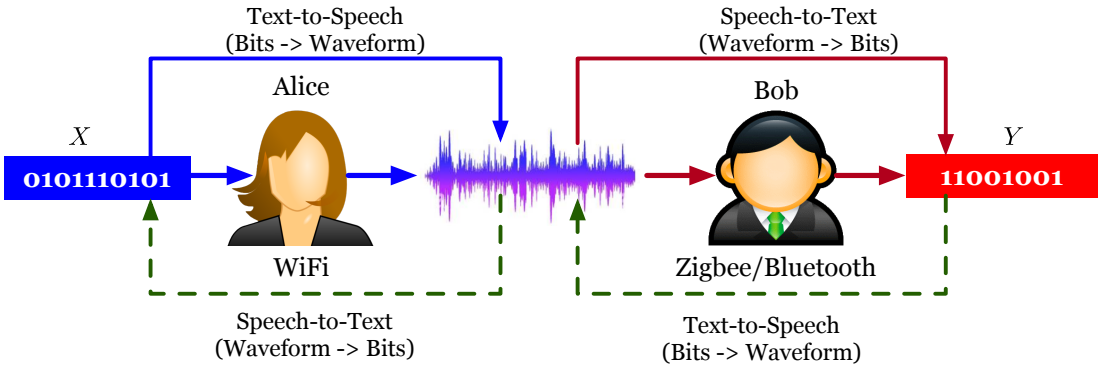


Figure 2.5: The task comparability between CTC and language translation.

2.2 System Design

2.2.1 Overview

Speech translation systems have been developed over the past several decades with the goal of helping people who speak different languages to communicate with each other. Such systems have usually been broken into three separate components: automatic speech recognition (i.e., speech-to-text, STT), machine translation (i.e., translating the transcribed text into the target language), and text-to-speech synthesis (TTS, i.e., generating speech in the target language from the translated text).

We can re-model the CTC problem from the perspective of NMT, as shown in Fig. 2.5. In the figure, a WiFi sender (i.e., Alice) and an IoT receiver (i.e., Bob) speak different languages. We deliver the source bits to Alice, who can generate the voice signal. This procedure is viewed as the TTS. Then the voice is captured by Bob, who makes an effort to understand the voice using his language by homophony. This procedure is viewed as the STT. Therefore, we can use a TTS and STT components to model the WiFi sender and an IoT receiver, respectively. The whole procedure is formalized as follows:

$$P_{\text{iot}} \approx \text{SST}_{\text{iot}}(\text{TTS}_{\text{wifi}}(P_{\text{wifi}})) \tag{2.1}$$

where P_{wifi} is the payload of a WiFi packet and P_{iot} is an entire IoT packet. The above

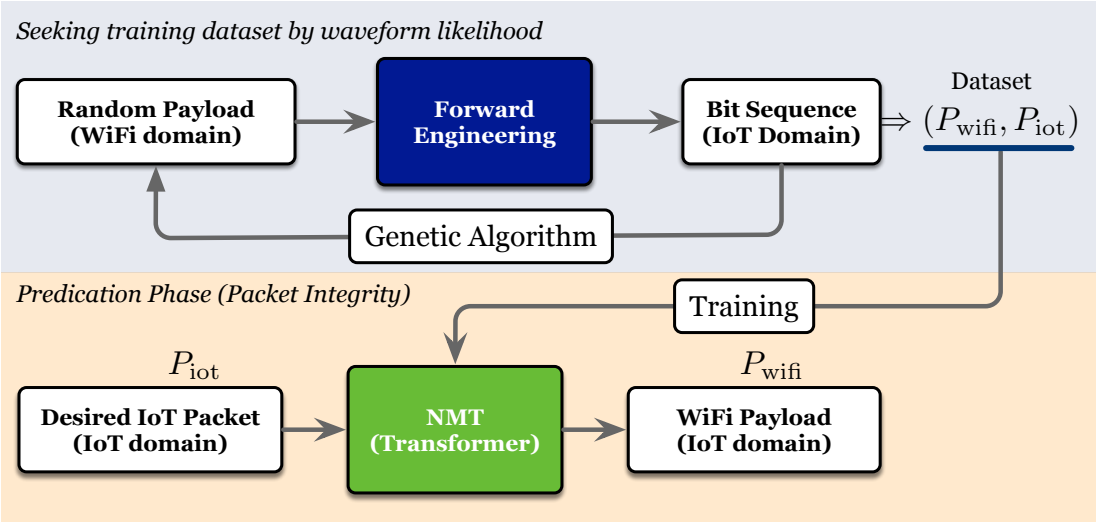


Figure 2.6: Architecture of XiTuXi. In the first phase, we use forward engineering to seek the training dataset (i.e., corpus). In the second phase, we train an NMT using the corpus for the purpose of translating an arbitrary IoT packet to a WiFi payload.

equation suggests that the heart of our problem is developing a language translation system for "translating" an IoT packet to a WiFi payload.

Inspired by such task comparability, we propose a general-purpose solution, XiTuXi, which can be applied to achieve the CTC from a WiFi sender to any type of IoT receiver. Compared with the previous reverse engineering-based approach, the NMT-based XiTuXi is purely data-driven without the need for knowing the communication details or manual intervention, as the current NLP does. The WiFi sender, the IoT receiver, and the in-between wireless channel together are viewed as a functional black box, which can be approximated by a neural network.

The architecture of XiTuXi is shown in Fig. 2.6. Specifically, it contains two phases:

- **Phase I: Building CTC Corpora**. At the heart of NMT is the training dataset (i.e., a corpus in NMT). To build a vast corpus database, we propose GA-based forward engineering to find out a sufficient number of mappings from WiFi payloads to IoT packets whose waveforms are highly similar. The details are elaborated in §2.2.2.
- **Phase II: Training NMT model**. We employ the well-known NMT model

called Transformer to resolve our problem. The Transformer takes advantage of the self-attention mechanism for sequence-to-sequence translation tasks while holding context sensitivity. We make efforts to adjust the input bit sequence for the Transformer in §2.2.3.

For clarity, we use the CTC between a WiFi sender (802.11n) and a ZigBee receiver (802.15.4) as an example to introduce our CTC solution in the subsequent sections. However, it can be easily extended to any other IoT protocol.

2.2.2 Phase I: Building CTC Corpora

A sufficient number of the well-annotated training dataset (i.e., corpora) is crucial for data-driven NMT. In this section, we introduce the method of forward engineering and the GA to drive forward engineering.

Forward Engineering

Following the settings in previous work [1], the length of a Zigbee packet is set to 32 bytes (256 bits), including a 25-byte payload and 7-byte header. Each ZigBee symbol represents 4 bits and lasts $16\mu s$. Thus, the duration of a Zigbee packet is $256/4 \times 16 = 1024\mu s$. On the contrary, each WiFi symbol lasts $4\mu s$, so a total of $1024/4 = 256$ WiFi symbols are available to emulate the waveform of a ZigBee packet. Considering 64-QAM modulation and 48 data subcarriers, each WiFi symbol represents $6 \times 48 = 288$ bits. Given a FEC rate of $\frac{3}{4}$, the payload of the WiFi packet should contain $256 \times 288 \times \frac{3}{4} = 55,296$ bits. Our ultimate goal is to find a bit sequence out of $2^{55,296}$ in the WiFi domain, which can be mapped to a bit sequence out of 2^{256} in the ZigBee domain. Formally, the mapping is defined as follows:

$$P_{\text{wifi}} \in \mathcal{P}_{\text{wifi}}^{2^{55,296}} \mapsto P_{\text{zigbee}} \in \mathcal{P}_{\text{zigbee}}^{2^{256}} \tag{2.2}$$

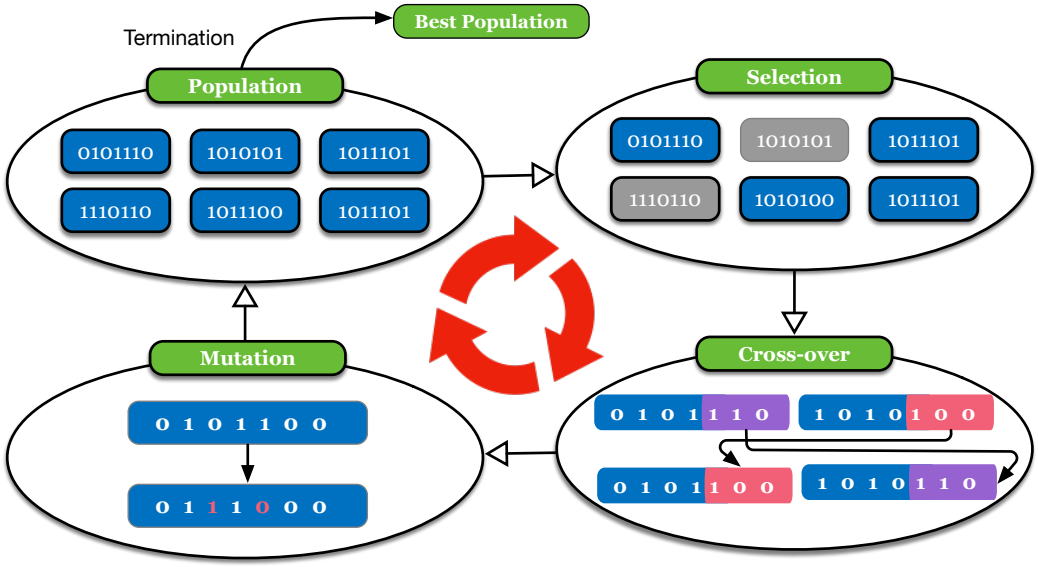


Figure 2.7: Workflow of GA

where $\mathcal{P}$ denotes the value space. Finding a particular map for a given P_{zigbee} (i.e., our ultimate goal) by brute force is harder than looking for a needle in a haystack because the combinations are more than the number of atoms in the universe. This is why previous reverse engineering is a tough and tedious task.

To address the above issue, we propose forward engineering to build the corpus. Specifically, we use a WiFi sender to transmit a random bit-sequence P_{wifi}. A ZigBee receiver is forced to receive and decode the OFDM(P_{wifi}) as P_{zigbee} using the OQPSK decoder. When OFDM(P_{wifi}) $\approx$ OQPSK(P_{zigbee}), we add the pair ($P_{\text{wifi}}, P_{\text{zigbee}}$) into the corpus; otherwise, they are dropped. ZigBee devices adopt OQPSK as their modulation scheme. This procedure treats the WiFi sender, channel, and ZigBee receiver as a black box. It repeats until a sufficient number of pairs are found. The procedure proceeds from the WiFi sender to the Zigbee receiver, so we call this approach *forward engineering*. It aims to pick a small percent of the mappings from the almost infinitely possible pairs to train an NMT.

Searching with GA

Forward engineering still faces a vast search space. We next use a heuristic algorithm called the GA to solve this problem. The GA is inspired by Charles Darwin' s theory of natural evolution. This algorithm reflects the process of natural selection, where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. Fig. 2.7 shows the algorithm workflow at a high level. The algorithm randomly generates a set of *initial chromosomes*, all of which are called *population*. Each chromosome is a bit sequence representing a possible solution to the problem. There are three evolution phases. The first phase is *natural selection*, in which the chromosomes are ranked using a fitness function that suggests how well each chromosome solves the problem with a score. The fittest chromosomes to the problem are selected for reproduction while the remaining are discarded. The second phase is *crossover*, during which two chromosomes are randomly selected to give birth to two new chromosomes. Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen randomly from within the genes. Offspring are created by exchanging parents' genes until the crossover point is reached. In the third phase of *mutation*, random bits of the offerings (i.e., genes) are flipped to create new chromosomes. The mutation helps maintain diversity within the population and prevent premature convergence. The above process is repeated until the algorithm has converged to meet goal criteria, e.g., new offspring are not significantly different from the previous generation.

The GA perfectly fits our demand for two reasons. First, it is particularly useful when the search space is discrete with many local maxima [11] as our scenarios, i.e., there are a large number of bit sequences in the WiFi domain, which can be recognized by a ZigBee receiver, and each is a local maximum. Second, the GA requires the solutions to be encoded in the format of bit sequences, whereas the WiFi payloads are naturally in the form of bit sequences without the need for encoding efforts. Next, we introduce how the GA is integrated into our problem:

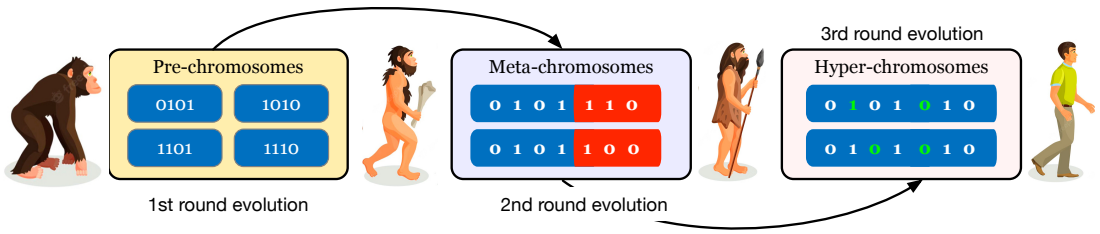


Figure 2.8: Progressive evolution.

(1) Initialization: The GA fixes the length of all chromosomes for reciprocal crossover, so we need to roughly estimate the length of the required WiFi payload, whose duration must be greater than that of the ZigBee packet of interest. The GA models each WiFi payload in fixed length as a chromosome and produces random chromosomes as the first generation. These chromosomes are fed into the WiFi physical layer for baseband processing (see Fig. 2.1). The modulation output is used for fitness evaluation.

(2) Fitness function: This function evaluates the performance of any given chromosome. A ZigBee receiver uses OQPSK for modulation. Specifically, the phase difference between two consecutive I/Q samples is computed to determine chip values with thresholding, i.e., outputting "0" or "1" if the shift is greater (or less) than zero degrees. After collecting 32 chips, ZigBee maps these chips into four bits according to the predefined symbol-to-chip codebook. Thus, given an arbitrary bit sequence in the WiFi domain, we can always obtain a decoded result anyway in the ZigBee domain. Let P_{wifi} and P_{zigbee} denote the transmitted WiFi payload and the decoded result, respectively. Formally, the two bit-sequences have the following relationship:

$$P_{\text{zigbee}} = \text{OQPSK}^{-1}(\text{OFDM}(P_{\text{wifi}}))$$

where $\text{OFDM}(\cdot)$ is the OFDM modulator and $\text{OQPSK}^{-1}(\cdot)$ is the OQPSK demodulator. Unfortunately, we have no idea about what results should be decoded out. No "ground truth" is available for evaluation. However, *we anticipate the waveform of decoded result is highly similar to the original OFDM waveform, so we can use the waveform likelihood as the fitness.* We reversely generate the waveform for the decoded P_{zigbee} using the ZigBee modulator denoted by $\text{OQPSK}(\cdot)$. The resulting

waveform is denoted by OQPSK(P_{zigbee}). The waveform similarity is defined as the normalized distance between the received waveform of the original WiFi bit-sequence and the reversely generated waveform as follows:

$$\mathrm{F} = d\Big(\mathrm{OQPSK}(P_{\text{zigbee}}), \mathrm{OFDM}(P_{\text{wifi}})\Big)$$

Notably, a less F score means higher similarity. The distance function $d(\cdot)$ depends on the modulation scheme that the IoT protocol adopts. For ZigBee protocol, the distance function is defined as follows:

$$d(s_1, s_2) = \frac{1}{N\pi} \sum_{n=2}^{N} \left| \arctan\left(s_1(n)s_1^*(n-1)\right) - \arctan\left(s_2(n)s_2^*(n-1)\right) \right| \tag{2.3}$$

where $s(n)$ is the n^{th} sample and $s(n-1)^*$ is the conjugate of $s(n-1)$, i.e., $n = 1, 2, \ldots, N+1$. We use the normalized absolute phase difference to indicate the waveform likelihood. We can also leverage the normalized amplitude difference as the fitness score for other keying schemes.

(3) Natural selection: We will go through multiple iterations of evolution. At the beginning of each iteration, we use the above function to evaluate each chromosome's fitness in the current population. We keep the top 60% chromosomes as parent chromosomes for the repopulation. **(4) Cross-over and mutation**. We randomly flip bits in the parent chromosome to produce a new chromosome for mutation. To give birth to a new chromosome, two parents are chosen randomly. The cross-over aims to preserve some portions of the chromosomes that can reflect the common features. This operation spontaneously fits our scenario where some fields (e.g., the preamble and delimiter) should remain and be inherited by all offerings. **(5) Convergence**. The convergence process is the final stage of the GA, where the population has reached a stable state and no further improvement is possible. The convergence process ensures that the algorithm has found a near-optimal solution for the CTC, and avoids wasting computational resources on unnecessary iterations. The GA stops once the best-performing chromosome reaches a fitness threshold or the maximum number of

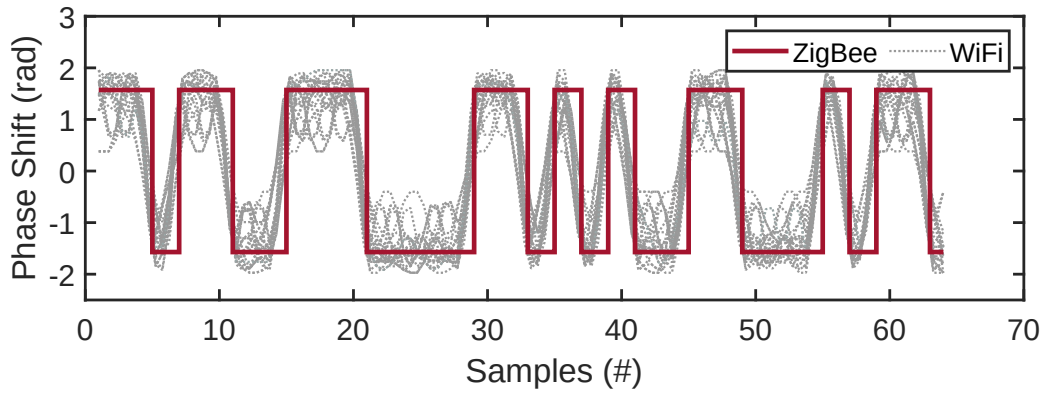


Figure 2.9: Examples of pre-chromosomes that emulate the Zigbee preamble.

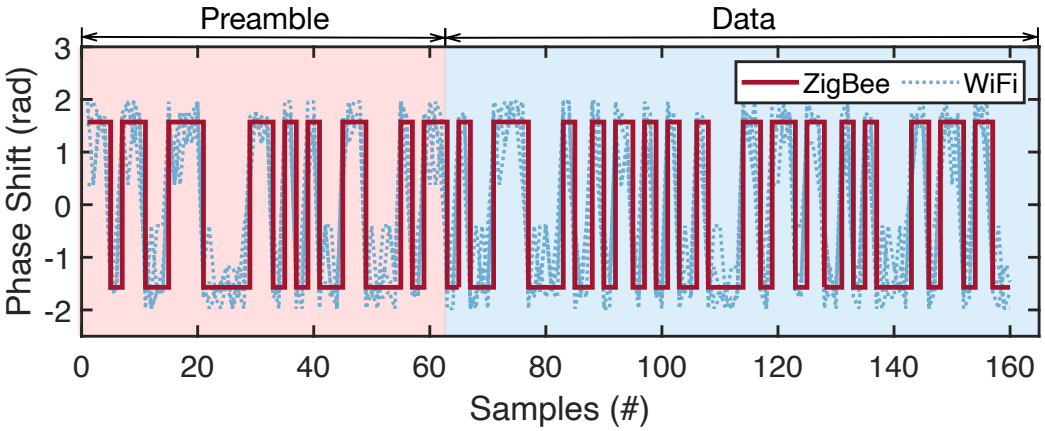


Figure 2.10: Examples of meta-chromosomes

iterations is reached.

Progressive Evolution

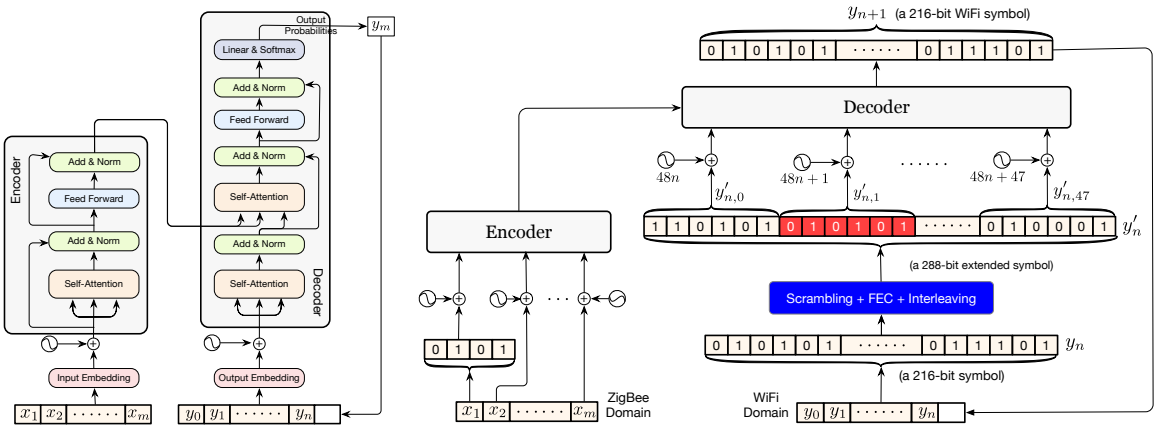
The GA generally starts from a set of randomly generated chromosomes. Our search space is huge, so the random initialization is insufficiently efficient. According to Charles Darwin's theory, the evolutionary grade is improved with a steady and progressive process, which allows creatures to evolve from simple to complex constitutions progressively. We also follow the progressive evolution by launching three main rounds of evolutions, as shown in Fig. 2.8.

(1) Pre-chromosomes: We start the first round of evolution by searching for the chromosomes, which can only emulate the 32-bit ZigBee preamble. These short chro-

mosomes are called *pre-chromosomes*. Random pre-chromosomes are produced as the initial generation. The preamble is predefined with a constant bit sequence (i.e., ground truth), so we can use the Hamming distance to evaluate their fitness. The GA converges when the best-performing pre-chromosomes all reach the threshold. Fig. 2.9 shows 20 pre-chromosomes that emulate the preamble.

(2) Meta-chromosomes: In the second round, we append random bits to the searched pre-chromosomes to create the initial meta-chromosomes. Namely, the preamble bits of the meta-chromosomes are inherited from pre-chromosomes, but the remaining bits are randomly generated. The length of the meta-chromosome is variable owing to the varying ZigBee payload. Thus, we can pre-train meta-chromosomes with different lengthes (e.g., 50, 100, 1000, ..., 80,000 bits), each of which requires an individual GA process because unequal-long chromosomes cannot take cross-over. However, the longer meta-chromosomes can be extended from the shorter meta-chromosomes found already, i.e., a 100-bit meta-chromosome is generated from a random 50-bit meta-chromosome plus 50 random bits. Meanwhile, the cross-over occurs in the new 50 bits. Such an incremental searching method can accelerate the coverage effectively because later bits do not affect the waveform of former bits. Fig. 2.10 shows a meta-chromosome (including the preamble and 20-bit payload) compared with the decoded result. Pre-chromosomes and meta-chromosomes can be decoded using a simulator without real communication to save time. When accepting a chromosome (i.e., a WiFi payload), the simulator modulates it using WiFi OFDM onto the carrier to generate an in-air waveform, which is later down-converted and sampled into the bit sequence for demodulation using ZigBee OQPSK. We evaluate their fitness by forcing a ZigBee simulator to decode them and compare the decoded result with the desired ZigBee packet. We employ chip error rate as an additional end-to-end metric, which can more directly reflect the fitness between two bit sequences.

(3) Hyper-chromosomes: With respect to the proprietary configurations of hard-



(a) Transformer Architecture (b) XiTuXi Architecture

Figure 2.11: Applying Transformer to XiTuXi. (a) shows the architecture of the Transformer; (b) shows how we adapt the input and output bit sequences to the requirement of the Transformer. Specifically, each ZigBee symbol is defined as a word as the input. a WiFi symbol is divided into 48 words after the FEC and interleaving, each of which is a 6-bit 64-QAM point. In each iteration, the Transformer is fed with 48 new words but predicts a single WiFi symbol.

ware, we start the third round of evolution, in which the previously searched meta-chromosomes are used as the first generation. The chromosomes are passed into the real WiFi sender and received by a real ZigBee receiver. The evolutionary meta-chromosomes that a ZigBee receiver can successfully decode are renamed hyper-chromosomes.

To amass sufficient data for the following deep learning model, the previously outlined stages go through multiple iterations. Each iteration aims to expand the dataset by collecting new pairs of hyper-chromosomes and ZigBee payloads. The process continues until a desired volume of data has been collected. This data collection stage serves as the foundation for building comprehensive datasets tailored to specific IoT protocols, ensuring the model's adaptability and effectiveness in diverse real-world scenarios.

2.2.3 Phase II: Neural Translation

The purpose of forward engineering is to find some pairs of WiFi bit-sequences and ZigBee bit-sequences, which are highly similar with respect to the waveform in the physical layer. Different from previous backward engineering, forward engineering can find some mappings but cannot tell the result for a desired IoT packet. Thus, we still need an NMT to discover the rationale behind the reverse procedure. In this section, we first introduce how we apply the Transformer to resolve our problem.

Background of Transformer

Sequence to Sequence (Seq2Seq) model is an NMT that transforms a given sequence of elements into another sequence. There are many Seq2Seq models for translation tasks, such as the LSTM [12], RNN [13], Transfomer [14], and so on. Generally speaking, Seq2Seq models consist of an encoder and a decoder. Particularly, the attention-based Seq2Seq has attracted plenty of attention from the community. In the attention mechanism, the encoder first decides the keywords and weighs words with different importance. Those new keywords make the translation much easier for the decoder because it knows which parts of the sentence are important and which key terms give the sentence context. Following this practice, we employ the famous attention-based NMT model called Transformer [14], which demonstrates great success in the Google Translator, GPT (Generative Pre-trained Transformer) and ChatGPT [15].

The architecture of the Transformer is shown in Fig. 2.11(a). The encoder is on the left, whereas the decoder is on the right. Both encoder and decoder are composed of modules that can be stacked on top of each other multiple times. Let $I = (x_1, x_2, \ldots, x_n)$ denote the original entire sentence and $O = (y_1, y_2, \ldots, y_{m-1})$ denote the currently translated result, where x_i and y_i represent the worlds in two different languages, respectively. The Transformer accepts both I and O as input and predicts the next word y_m. The output probabilities are a high-dimensional probabil-

ity vector, each of which corresponds to the probability of a word in the dictionary. The word with the highest probability is selected as the new word y_m. Once y_m is obtained, the O is updated to $(y_1, y_2, \ldots, y_{m-1}, y_m)$ and then re-fed into the network to predicate the next world y_{m+1}. Initially, O is empty and this process is iterated until no new word is predicted. Clearly, the Transformer considers not only the original sentence but also the current translating context to choose the next most appropriate word. Owing to the space limit, we encourage readers to refer to [14] for details.

Adopting Transformer for CTC

Our goal is to find an appropriate WiFi bit-sequence for the given ZigBee bit-sequence, i.e., translating a bit sequence in the ZigBee domain to one in the WiFi domain. We maintain the architecture of the Transformer but update the input and output. The word embedding is unnecessary because the input and output are bit sequences already. However, the bit sequences must be appropriately segmented to be adaptive to the requirement of the Transformer.

(1) Encoder input. The input of the Transformer's encoder is the bit sequence in the ZigBee domain. As shown in Fig. 2.11(b), we view each ZigBee symbol as a *word*. Each symbol contains four bits so we segment the bit sequence into words every four bits. Each word is a four-dimensional vector. These words $\{x_1, x_2, \ldots, x_m\}$ and their encoded positions $\{E(1), E(2), \ldots, E(m)\}$ are fed into the encoder.

(2) Decoder input. The input of the Transformer's decoder is the WiFi symbols found for the emulation currently. Each WiFi symbol contains 48 QAM points, in which only the QAM points piggybacked at the ZigBee channel can be received by the ZigBee receiver. As shown in Fig. 2.11(b), the six bits (64-QAM) highlighted in red are modulated onto the second subcarrier, which the ZigBee receiver listens at. We want the Transformer to focus on this QAM point by using the attention mechanism because it is the key to emulating the waveform. The remaining QAM points at other subcarriers are free to choose. If a whole WiFi symbol is used as a word,

the self-attention mechanism is underachieved because the Transformer processes the sentence word by word. To address this issue, we first segment the input bit sequence into *words* on the decoder side. Specifically, let $\{y_0, y_1, \cdots, y_n\}$ denote the n original WiFi symbols that we have obtained already so far from the decoder. Given a channel coding rate of 3/4, each original symbol contains 216 bits ($= 288 \times \frac{3}{4}$). Now, each 216-bit symbol y'_n s extended to a 288-bit coded symbol denoted by y'_j after the scrambler, FEC and interleaver. Further, these 288 bits are segmented to 48 *words* every 6 bits, each of which corresponds to a 64-QAM point. We use $\{y'_{n,0}, \ldots, y'_{n,j}, \ldots, y'_{n,47}\}$ to denote the words segmented from the coded symbol y'_n. The corresponding position of word $y_{n,j}$ is updated to $48n + j$. The sum of $y_{n,j}$ and $E(48n + j)$ is eventually fed into the Transformer's decoder. Note that the scrambling, FEC and interleaving are the pre-processing tasks taken by our deterministic algorithm instead of the real device, with three configuration parameters of scrambler seed, FEC coding rate and permutation order. These parameters are known in the WiFi standard. If they are inaccessible, these pre-processing modules can be simply replaced by a fully-connected network that is trained together with the post-decoder.

(3) Decoder output: The output of the decoder is the probability vector across the dictionary. Unfortunately, we want the decoder to output an original WiFi symbol, resulting in a 2^{216}-scale dictionary. To address this scalability issue, we change the output vector to indicate the probabilities of 216 bits. Specifically, we change the size of the output vector to 216. Each item indicates the probability of bit one. If the probability is greater than 0.5, the corresponding bit is 1; otherwise, it is a bit 0. The loss function of the decoder should be updated to compute the Hamming distance between the output bit vector and the ground truth.

Addressing Packet Integrity

A valid packet must meet some requirements, such as the packet structure, CRC validation, etc. We call this requirement *packet integrity*. Forward engineering and

Transformer are oriented to bit-sequences, so both cannot guarantee the packet integrity but ensure the waveform similarity. However, the application knows exactly what ZigBee packets are desired to be received. Thus, the application can provide the entire packet P_{zigbee} that meets the integrity constraint. The Transformer translates P_{zigbee} to a bit sequence P_{wifi} in the WiFi domain. Once the WiFi sender transmits P_{wifi}, the ZigBee receiver can exactly receive P_{zigbee}, which definitely meets the integrity constraint because it is constructed by the application. On the other hand, the packet P_{wifi} is passed to the PHY. The WiFi sender assembles it as payload and adds the corresponding header, footer and CRC automatically. However, the ZigBee receiver ignores these fields because they do not contain the ZigBee preamble. In short, the packet integrity must be confirmed by the application.

2.2.4 Implementation

We have implemented 30 CTC combinations from 10 different WiFi configurations to four IoT devices.

• **WiFi configurations**. Table 2.1 lists all possible configurations of WiFi in accordance with all versions available at 2.4 GHz, including 802.11b, g, n, and ax. Considering that some IoT devices prefer to work at 800-900 MHz ISM bands for better penetration, we also introduce the 802.11ah, which uses 915 MHz license-exempt bands to provide extended-range Wi-Fi networks. 802.11ac is not considered because it works at 5GHz. In addition to 16-QAM and 64-QAM, we are the first to test the 256-QAM regarding 802.11n and 802.11ax. The duration of a WiFi symbol varies in $0.72\mu s$, $3.6\mu s$, $4\mu s$, $13.6\mu s$, $14.4\mu s$, $16\mu s$ and $40\mu s$.

• **IoT configurations**. Table 2.2 lists the five types of IoT techniques, namely, ZigBee@2.4 GHz, BLE@2.4 GHz, LoRa@2.4 GHz, LoRa@915 MHz, and Sigfox@915 MHz. The modulations involve OQPSK, GFSK, CSS, and DBPSK. The symbol duration varies from $1\mu s$ to $10,000\mu s$. Usually, the protocols operating at 915 MHz

Table 2.1: Configurations of WiFi senders

#	Protocol	Modulation	BW	Duration	Freq.
W1	802.11b	CCK	22MHz	0.72μs	2.4GHz
W2	802.11g	BPSK	20MHz	4μs	2.4GHz
W3	802.11n	QPSK	20MHz	4μs	2.4GHz
W4	802.11n	16QAM	20MHz	4μs	2.4GHz
W5	802.11n	64QAM	20MHz	4μs	2.4GHz
W6	802.11n	64QAM	20MHz	3.6μs	2.4GHz
W7	802.11ax	256QAM	20MHz	13.6μs	2.4GHz
W8	802.11ax	256QAM	20MHz	14.4μs	2.4GHz
W9	802.11ax	256QAM	20MHz	16μs	2.4GHz
W10	802.11ah	64QAM	20MHz	40μs	915MHz
W11	802.11ah	256QAM	20MHz	40μs	915MHz

Table 2.2: Configurations of IoT receivers

#	Protocol	Modulation	BW	Duration	Freq.
IoT1	ZigBee	OQPSK	2MHz	16μs	2.4GHz
IoT2	BLE	GFSK	1MHz	1μs	2.4GHz
IoT3	LoRa	CSS	500KHz	256μs	2.4GHz
IoT4	LoRa	FSK	500KHz	2μs	915MHz
IoT5	Sigfox	DBPSK	192KHz	10,000μs	915MHz

define much longer symbol duration because the wireless period is longer than that at 2.4 GHz.

• **CTC combinations**. Table 2.3 lists the tested 30 CTC combinations. The remaining 25 combinations either operate at different frequencies or are covered already. The combinations of T1-T9, T10-T18, T19-T28, and T29-T30 are the communications from WiFi to ZigBee, to BLE, to LoRa, and to Sigfox, respectively. "Scale" is the scale of corpus sought via the GA-enabled forward engineering. "Previous" indicates if the combination has been achieved by any previous work. For each CTC combination, we need to train an individual Transformer model.

• **Machine learning**. We used the Matlab WLAN toolbox, Bluetooth toolbox, and ZigBee Toolbox to implement WiFi, Bluetooth, and ZigBee protocol stacks, respec-

Table 2.3: Achieving CTC across WiFi senders and IoT receivers

#	CTC	IoT	Scale	Past Works	Train (h)	Test (s)
T1	W1→ IoT1	ZigBee	40K	×	53.1	0.33
T2	W2→ IoT1	ZigBee	38K	×	60	0.37
T3	W3→ IoT1	ZigBee	35K	×	61.5	0.4
T4	W4→ IoT1	ZigBee	35K	×	74.3	0.62
T5	W5→ IoT1	ZigBee	32K	WEBee [1] LongBee [2] TwinBee [3] WIDE [7]	91.17	0.91
T6	W6→ IoT1	ZigBee	30K	×	93.02	1.1
T7	W7→ IoT1	ZigBee	20K	×	106.2	1.83
T8	W8→ IoT1	ZigBee	20K	×	105.7	2.05
T9	W9→ IoT1	ZigBee	25K	×	103.33	1.67
T10	W1→ IoT2	BLE	40K	WiBeacon [4]	49.5	0.29
T11	W2→ IoT2	BLE	40K	×	53.67	0.31
T12	W3→ IoT2	BLE	36K	×	55.8	0.36
T13	W4→ IoT2	BLE	38K	×	66.2	0.5
T14	W5→ IoT2	BLE	35K	BlueFi [5] BlueFi2 [17]	88.34	0.72
T15	W6→ IoT2	BLE	32K	×	92.71	0.89
T16	W7→ IoT2	BLE	20K	×	102.9	1.25
T17	W8→ IoT2	BLE	20K	×	104.6	1.44
T18	W9→ IoT2	BLE	25K	×	100.83	1.21
T19	W1→ IoT3	LoRa	35K	×	51.7	0.35
T20	W2→ IoT3	LoRa	30K	×	56.33	0.4
T21	W3→ IoT3	LoRa	30K	×	57.5	0.46
T22	W4→ IoT3	LoRa	28K	×	62.1	0.67
T23	W5→ IoT3	LoRa	25K	×	75.4	0.99
T24	W6→ IoT3	LoRa	25K	×	82.52	1.16
T25	W7→ IoT3	LoRa	16K	×	96.19	1.88
T26	W8→ IoT3	LoRa	18K	×	96.67	2.1
T27	W9→ IoT3	LoRa	20K	WiRa [6]	98.46	1.83
T28	W10→ IoT4	LoRa	25K	×	105.3	1.86
T29	W10→ IoT5	Sigfox	20K	×	99.01	1.52
T30	W11→ IoT5	Sigfox	20K	×	102.75	2.23

tively. The LoRa is implemented using the open source project [16]. These toolboxes are employed to find the appropriate corpora. The GA algorithm is also implemented using the Matlab API. We use the PyTorch framework to develop the Transformer and run it on a Linux server with an AMD 5900x (4.9 GHz) processor, 64 GB RAM, and two NVIDIA RTX3080Ti GPUs. The Adam optimizer with a learning rate of 10^{-5} is used to update the network parameters. The training converges well after 500 training epochs.

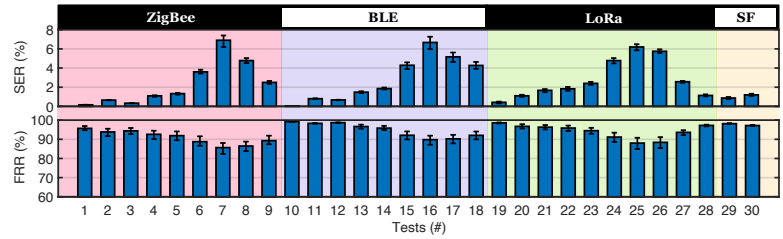


Figure 2.13: Accuracy across CTC combinations. The detailed configurations for Test 1- Test 30 refer to Table 2.3.

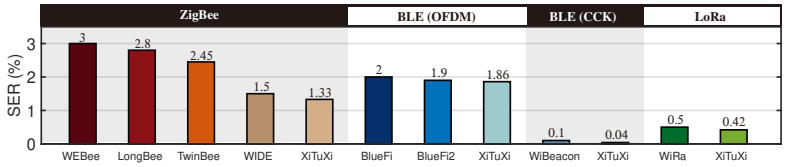


Figure 2.14: Comparison with SOTA regarding the physical-layer performance

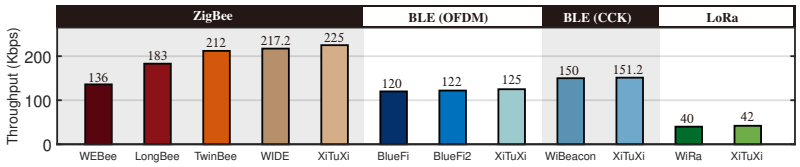


Figure 2.15: Comparison with SOTA regarding the link-layer throughput

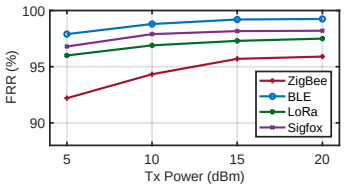


Figure 2.16: FRR vs. Tx Power

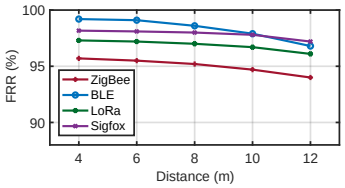


Figure 2.17: FRR vs. Distance

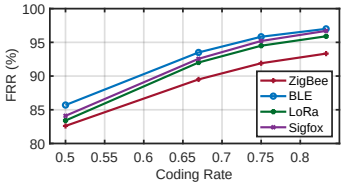


Figure 2.18: FRR vs. Coding Rate

2.3 Evaluation

We start with benchmark experiments by using the USRP platform to provide insights into the working of XiTuXi.

Experimental Setup. We use USRP platforms as the WiFi transmitter and an IoT receiver to evaluate the performance of the 30 CTC combinations in our office environment, where the 2.4GHz and 915MHz are very crowded. At least five nearby APs at 2.4GHz and two 5G-base stations at 915MHz are operating on the same WiFi channel. By default, the transmitting power is fixed at 20dBm; the distance is four meters. The WiFi payloads are generated by the Matlab Toolbox and are broadcasted by the RF frontend (i.e., USRP). Another USRP platform is used to receive and downconvert the packets using an IoT protocol. We also use the Matlab Toolbox to decode the packet offline in the physical layer. Such flexible experimental

platforms allow us to conduct the benchmark on a large scale quickly.

2.3.1 Overall Accuracy

We evaluate the accuracy of XiTuXi from the physical and link layers, respectively. The results are shown in Fig. 2.13. From the figure, we have the following findings:

• **Physical-layer performance**. In the physical layer, we employ the SER, computed as the ratio of the flipped bits in a packet. XiTuXi achieves an overall mean SER of 2.55% across the 30 protocol combinations. The mean SERs for ZigBee (T1-T9), BLE (T10 -T18), LoRa (T19-T28) and Sigfox (T29-T30) are 2.37%, 2.8%, 2.78% and 1.04%, respectively. Exceptional SER rises are observed in T6-T8, T15-T17, and T24-T26, in which the WiFi symbol duration varies from 3.6 - 14.4μs. These irregular durations result in misalignment in the symbol level, making the emulation context-dependent. The corresponding difficulty is reflected from the 3.5× higher SERs in these combinations than others. However, previous CTC systems simply fail to deal with the misalignment.

• **Link-layer performance**. In the link layer, we employ the FRR, computed as the ratio of the successfully received packets, which must pass the CRC validation. The FRR focuses on the integrity of packets. Even if a few bits are flipped, the packet may still be successfully decoded because of the correctness. We assemble intact IoT packets with a minimum of 25 bytes for testing. To ensure statistical validity, we obtain the average results of 1000 packets for each setting. XiTuXi achieves an overall mean FRR of 93.6%. In terms of the IoT protocols, the mean FRRs for the four IoT protocols are 91%, 94.8%, 94.05%, and 97.69%. The trend remains consistent with the SER, i.e., higher SER means a lower FRR because more flipped bits are vulnerable to recovery failure. Moreover, the results also suggest that the FRR decreases as a function of the order of QAM. Higher-order QAM configurations of WiFi lead to relatively lower FRR because they embed more bits in a single symbol, which increases

the emulation difficulty.

• **Summary**. In short, our results fully demonstrate the potential of CTC by using WiFi to emulate the waveforms of all kinds of popular IoT protocols with high accuracy. They also verify the feasibility and effectiveness of using XiTuXi as a general-purpose solution to solve this problem perfectly.

2.3.2 Comparison with SOTA Works

Next, we compare XiTuXi with the state-of-the-art (SOTA) works for ZigBee, Bluetooth, and LoRa. For fairness, we evaluate XiTuXi using the identical configurations as the past CTC systems [2]. The Sigfox protocol is not considered here due to the absence of related work.

• **Physical-layer performance**. We still employ the SER to evaluate the physical layer performance. The results are shown in Fig. 2.14. We have the following findings. **(1) ZigBee**: WEBee [1], LongBee [2], TwinBee [3], WIDE [7] and XiTuXi adopt the T5 configuration (see Table 2.3) for the CTC from WiFi to ZigBee, and the past four systems achieve the mean SERs of 3%, 2.8%, 2.5%, and 1.5%, respectively. By contrast, XiTuXi achieves the mean SER of 1.33%, which further reduces the symbol errors by 55%, 51%, 46.8%, and 11.3%. Particularly, WEBee is the pioneering work that proposed the waveform-emulated CTC. Our results show that XiTuXi can further reduce the Hamming distance by 50% over WEBee. The outperformance of XiTuXi over WEBee or others is in the context-sensitive emulation, which does not limit to a few numbers of symbol replacements. XiTuXi breaks the limitation of an expert's practical experience in reverse engineering. While WIDE is a noteworthy system with best performance among past works, XiTuXi can surpass its capabilities. This is because WIDE relies on greedy algorithms, which, though effective, have a tendency to identify local optimal solutions rather than the desired global optimal

[2]Due to the lack of raw data, the results of SOTA systems are reproduced from the corresponding report.

solution. In contrast, XiTuXi employs a combination of GA and deep learning models, enabling it to consistently discover the global optimal solution. This distinction in methodology underscores the superior performance of our system in comparison to WIDE. **(2) BLE**: BlueFi [5] and BlueFi2 [17] adopt the T14 configuration (OFDM/WiFi), whereas WiBeacon [4] adopts the T10 configuration (CCK/WiFi). They achieve the mean SERs of 2%, 1.9%, and 0.1%. By contrast, XiTuXi has a mean SER of 0.04%, which further reduces error bits by 98%, 97%, and 60% compared with the three systems. WiBeacon exhibits an excellent performance than BlueFi and BlueFi2 because it adopts the CCK/WiFi, which allows the manipulation of phase shifts that perfectly emulate BLE chips [4]. The results of XiTuXi also verify the easiness by using CCK to emulate the BLE than OFDM, which is reflected in the 97% decline of SER. However, CCK is a very legacy WiFi protocol that is almost obsolete and thereby, its usage scenarios might be limited. **(3) LoRa**: WiRa [6] is the only past work that targets the CTC from WiFi to LoRa by using the T27 configuration. WiRa and XiTuXi achieve comparable mean SERs, i.e., 0.5% and 0.42%, respectively.

• **Link-layer performance**. We use the throughput to evaluate the link-layer performance. The throughput is defined as the number of bits successfully received per second. The throughput is not only determined by the FRR but also depends on the number of IoT packets embedded inside a WiFi packet because the header and the footer fields of WiFi are overheads. For fairness, we only embed a single IoT packet in a 1ms WiFi packet for all CTC systems. The throughput is computed across 1000 emulated packets (i.e., duration is 1s).**(1) ZigBee**. The maximum data rate of ZigBee is 250kbps. Actually, the WeBee, LongBee, TwinBee, and WIDE achieved 136kbps, 183kbps, 212kbps, and 217.2kbps, respectively. The results are consistent with the physical-layer performance evaluation. The TwinBee becomes closer to WIDE because it improves the throughput using a chip recovery technique. In contrast, XiTuXi achieves 225 kbps throughput without the need for recovery mechanisms. This remarkable achievement surpasses the performance of the four previously mentioned

systems by 89 kbps, 42 kbps, 13 kbps, and 7.8 kbps, respectively. **(2) BLE.** The maximum data rate of BLE is 1Mbps. Unlike the ZigBee, the duration of a BLE symbol is $< 1/4$ of a WiFi symbol (i.e., $4\mu s$), which means it takes $4\times$ longer to transmit an emulated BLE symbol than a real one even if a single WiFi symbol is used. Thus, the practical maximum data rate of emulated BLE should be less than 200kbps when considering other overheads. BlueFi, BlueFi2, and XiTuXi (OFDM) achieve comparable throughputs of about 120kbps. WiBeacon and XiTuXi (CCK) achieves 150kbps due to the lower SER. The throughput can be increased by $5\times$ (i.e., to 700kbps) via embedding at most five BLE packets inside a single WIFi packet. **(3) LoRa**. The maximum data rate of LoRa with FSK modulation is 50kbps. WiRa and XiTuXi achieve about 40kbps and 42kbps, respectively.

• **Summary**. XiTuXi significantly improves the CTC performance compared with the SOTA works regardless of which IoT protocol is targeted. Such outperformance is mainly attributed to the following reasons. First, forward engineering provides a huge volume of the training set, making the symbol selection more flexible than the previously proposed symbol-by-symbol replacement strategy. Second, the symbol selection is context-dependent, which facilitates the Transfomer to choose the better WiFi symbols by regarding the existing ones. Third, the powerful self-attention mechanism of the Transformer would choose the key symbols for the emulation but lower the priority of other unimportant symbols, which was never considered by reverse engineering.

2.3.3 Training Cost

Next, we evaluate the training cost associated with the development of the neural machine translation model for various protocol combinations. A detailed comparison of these training costs is presented in Table 2.3. Training times vary across different protocol combinations, spanning from 49.5 hours to 106.2 hours. On average, training times for ZigBee, BLE, LoRa, and Sigfox are 83.15 hours, 67.97 hours, 78.22 hours,

and 100.88 hours, respectively. These durations underscore the impact of the inherent complexity and difficulty involved in achieving CTC for diverse IoT protocols. As a general trend, training costs tend to increase in tandem with the dataset size, as the model processes a greater number of packets. One potential avenue for reducing training time is through the adoption of more powerful computing hardware, such as advanced GPUs.

2.3.4 Inference latency

We conclude our evaluation by examining the inference latency, which denotes the time required to generate the WiFi payload for a given IoT packet utilizing the trained model. Inference is conducted on a standard PC equipped with a single NVIDIA RTX 3080Ti GPU. To measure the inference cost for each CTC combination, we feed 1000 target packets into the respective model and calculate the average inference time. The results are shown in the Table 2.3. We have the following findings: (1) The average inference time of ZigBee, BLE, LoRa, and Sigfox are 1.03s, 0.77s, 1.17s, and 1.875s, respectively. The latency may not be suitable for applications requiring rapid response time. (2) The inference time cost ranges from 0.04s to 2.05s. This time difference can be attributed to the varying complexity of different modulation schemes. For example, the inference time needed for T9 is higher than that for T1, since W9 utilizes the more complex 802.11ax with OFDMA technology. In contrast, W1 employs 802.11b with CCK, which boasts simpler and less diverse waveforms.

2.3.5 Real-world Evaluation

Finally, we evaluate the performance of XiTuXi with commercial wireless devices and real-world applications. We conduct experiments using the commodity devices in Fig. 2.19, where the WiFi AP pre-loaded with OpenWrt [18] (GL-AR750S [19]) is used for transmitting the WiFi frame generated by XiTuXi across four protocols. XiTuXi

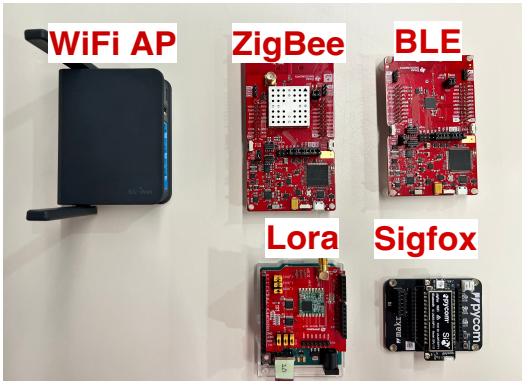


Figure 2.19: Experiment Device Setup

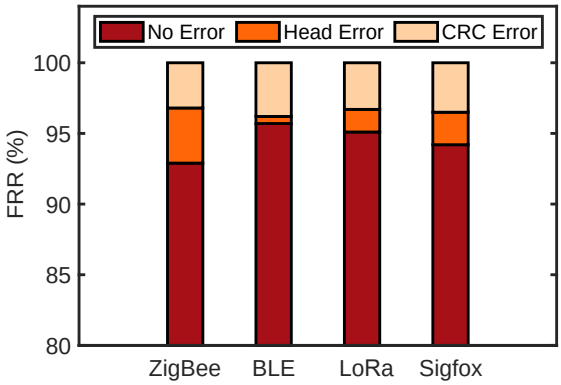


Figure 2.20: FRR vs. Protocols

does not use any OpenWrt-specific features. We use TI development boards with CC1352P [20] and CC2652R [21] wireless MCU as ZigBee and Bluetooth transceivers. The commercial LoRa platform equipped with Semtech SX1280 chip [16] and PyCom Sipy Sigfox board [22] is employed to receive the LoRa and Sigfox data. They are tested using the T5, T14, T28, and T30 protocol combinations.

Performance on Real Hardware

The performance of XiTuXi on the real commercial devices is shown in Fig. 2.20. We achieve mean FRRs (i.e., percent of no error) of 92.9%, 95.7%, 95.1% and 94.2% on the four IoT devices. The results remain consistent with the benchmark, i.e., about ±2% acceptable difference. We also employ a sniffer (i.e., USRP) to analyze the distributions of errors. The figure suggests the maximal CRC error of 3.9% among the four protocols, which almost accounts for the majority of its errors. Unlike other protocols, which adopt 2-byte CRC, BLE uses 3-byte CRC, which encounters a higher error probability. Similarly, ZigBee has major errors (i.e., 2.8% out of 3.2%) in the preamble because it adopts the longest preamble with 4 bytes. This experiment fully demonstrates the feasibility and effectiveness of XiTuXi on commercial devices and the similar performance as the benchmark.

Impact Analysis

Next, we measure the accuracy with respect to different system settings. **(1) Impact of TX power**. Fig. 2.16 shows the FRR as a function of the transmitting power of the WiFi sender regarding the four IoT protocols. In the figure, they show a similar trend, i.e., the FRR increases along with the transmitting power across all kinds of protocols. This is because higher TX power increases the signal-to-noise ratio, which facilitates encoding. The results also show that ZigBee requires a higher demand on the TX power for decoding compared with other low-power IoT protocols. By default, a WiFi AP is set up with a 20dBm transmitting power. In this setting, XiTuXi does not introduce noticeable FRR diversity. **(2) Impact of distance**. Fig. 2.17 compares the FRR of XiTuXi at different locations in a hallway. XiTuXi achieves higher accuracy (e.g., 97% at 4 and $>$ 94% at 12m) because the received signal strength of IoT packets is 6 dB higher on average at 3m than that at 12m, which is mainly caused by the channel attenuation and the multiple-path effect. **(3) Impact of coding rate**. WiFi adopts FCC to increase data reliability, i.e., inserting redundant bits into the data stream. The coding rate is the percent of how much of the real bits are transmitted. A higher coding rate means less redundancy. Fig. 2.18 shows the FRR increases as an increased coding rate (e.g., 84%@1/2 rate$\rightarrow$95%@5/6 rate). It is reasonable because the redundancy computed by the WiFi chip is out of the control of XiTuXi, which only feeds raw payload into the chip. A higher coding rate means more transmitted bits can be manipulated, so the waveform is more controllable.

Application Study

To demonstrate the practicality of XiTuXi, we employ the devices to transmit a picture in the format of JPEG from the WiFi AP to various IoT devices. The total file contains 8,516 bytes. For fairness, we assemble each packet with a fixed length (i.e., 20-byte payload) across different solutions. Thus, the image should be transmitted

using 426 data packets in total. The transmission is unidirectional, without acknowledgment and retransmission. The received results are shown in Fig. 2.2. JPEG encodes data in the frequency domain, so the packet reception errors cause the loss of some frequency components, making the picture become blurry. Notably, only XiTuXi can still receive more clear images than STOA works. XiTuXi works to receive the WiFi packet with 3.6μs-short CP. By contrast, WEBee and BlueFi received nothing.

2.4 Discussion

In this section, we discuss the limitation of XiTuXi and the practical issues that may happen during the usage.

2.4.1 Computational Time

Unlike traditional reverse engineering works, XiTuXi employs a deep learning architecture to automatically solve the CTC problems. However, the current system requires powerful GPUs and hundreds of hours to train. This limitation is particularly relevant for IoT applications that require immediate action. To speed up inference, we may employ advanced strategies such as asynchronous processing, model parallelism, and hardware acceleration in future work.

2.4.2 Deployment on Commerical Devices

One significant limitation is the current system's reliance on a large transformer model that demands substantial computational resources. This model complexity precludes deployment on edge IoT devices with strict hardware constraints, such as commercial Wi-Fi routers. In future work, we consider designing a more efficient mobile translation model tailored for deployment on commercial Wi-Fi routers. We can achieve this by exploring advanced model compression techniques such as quantization compres-

sion [23] and structure optimization [24, 25]. This can bridge the gap between our current research findings and practical, real-world applications, ultimately enhancing the feasibility and effectiveness of CTC strategies in the realm of network protocols.

2.4.3 Full-duplex CTC

At present, our system exclusively addresses CTC challenges from high-speed IoT devices (e.g., WiFi) to low-speed IoT devices (e.g., ZigBee and Bluetooth). However, enabling CTC support in the reverse direction, from low-speed devices to high-speed ones, remains an open problem. Existing research ZigFi [26] demonstrates that we can manipulate Zigbee devices to transmit packets over WiFi, disrupting channel state information to send data. In our forthcoming work, we intend to delve further into this direction, aiming to expand XiTuXi's capabilities to support the full-duplex CTC.

2.5 Related Work

Our work is broadly related to the research on CTC and natural language processing.

• **CTC**: Recently, CTC technology has received considerable attention from the wireless community [27, 1, 28, 29, 7, 30] and is experiencing numerous attempts. Basically, CTC technologies have been developed from the packet level to the symbolic level. At the packet level, past works enable CTC by changing packet length [27, 31], packet timing [28], packet sequence patterns [32, 33] and packet energy [29]. The drawback of these works is poor throughput. The symbolic level emulation is then explored to achieve higher throughput communication [1, 34, 2, 7, 3, 35, 36]. For example, WEBee [1] first achieves high-throughput CTC between WiFi and Zigbee via physical-level reverse engineering. The technical idea is also used for creating Zigbee→Bluetooth CTC [34], WiFi→Bluetooth CTC [5] and even LoRa→ZigBee

CTC [37]. However, all past works rely on empirical knowledge to find the link possibilities between heterogeneous technologies. Therefore, they cannot fully catch up with the rapid development of new wireless protocols. Our work addresses this dilemma by first introducing deep learning technologies to solve reverse engineering.

• **NMT**: NMT is a hot topic in the field of natural language processing, which has gained plenty of attention owing to its ability to produce high-quality translations [38]. Recent NMT researchers have proposed numerous neural network translation frameworks, including DNN-based NMT [39, 40], NMT with attention mechanism [41] and fully attention-based NMT [14, 42]. Currently, fully attention-based models, like Transformer [14], BERT [43], GPT [44], achieve the SOTA performance and dominate the community. Our work is inspired by these translation models for the natural language, but we extend them to solve the wireless signal emulation problem for creating CTC. This is the first time that the network data packets can be translated across protocols by using an NMT model.

2.6 Conclusion

This work breaks the traditional CTC research mindset of relying on expert empirical knowledge. We give a new perspective on CTC research by considering it as a translation task between network protocols and propose a novel neural translation model to find CTC strategies automatically. This work does not raise any ethical issues.

Chapter 3

Scaling Mobile IoT with eSIM

The Mobile Internet of Things harnesses cellular technologies to maintain reliable connectivity as devices traverse diverse network coverage areas [45]. It utilizes existing mobile networks, especially those tailored for reduced power use and broader coverage, such as LTE-M (LTE for Machines) and NB-IoT (Narrowband IoT). These technologies support a variety of applications, including vehicle telematics, fleet management, wearable health devices, and asset tracking, by offering cost-effective, efficient, and dependable communication over cellular networks.

The Subscriber Identity Module (SIM) acts as the ”key” for connecting to cellular networks. However, traditional removable SIM cards significantly hinder IoT development due to two main challenges: First, IoT environments are less controlled than those for mobile phones, making traditional SIMs prone to damage. Second, physically accessing IoT devices to replace SIM cards presents substantial logistical difficulties. To address these challenges, the GSM Association released new standards for a digital version of SIMs known as eSIMs (Embedded SIMs). This technology integrates physical SIM chipsets within the devices themselves, while network operator profiles are managed remotely via cloud services. This eSIM ecosystem provides the capability for industry stakeholders to remotely push, activate, deactivate, and re-

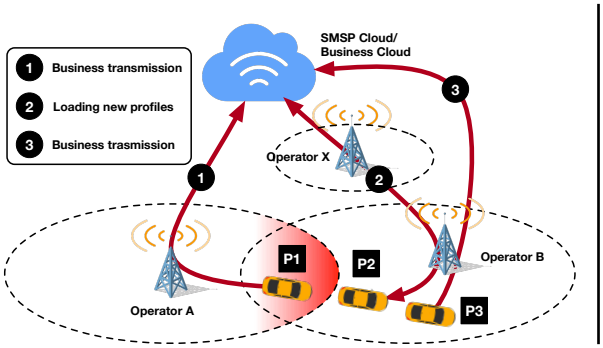
move operator profiles, thereby enhancing the flexibility and efficiency of managing IoT device connectivity. This shift not only simplifies the logistical challenges but also broadens the operational resilience and scalability of IoT networks.

The inception of eSIM technology dates back to the early 2010s, with the GSMA formalizing its initial specifications in 2013. Despite its growing use, the performance of eSIM-enabled mobile IoT has scarcely been studied or empirically assessed over the past decade. Assessing the usability and connectivity of eSIMs on a large scale presents significant challenges. Firstly, network operators have little motivation to promote eSIM adoption due to potential traffic loss and resultant revenue reductions. Secondly, conducting comprehensive real-world measurements on IoT devices is costly and complex; while controlled laboratory experiments can offer some insights, they often fail to capture the nuanced characteristics critical for detailed analysis.

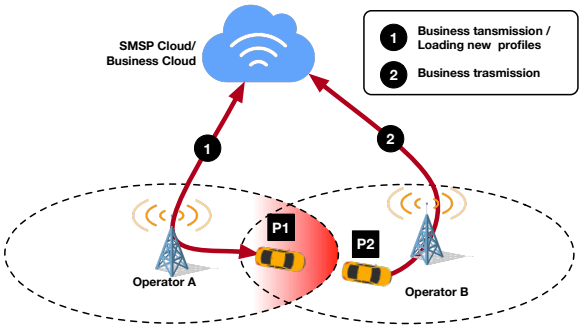
Measurement. To close the knowledge gap, we have partnered with a Subscription Management Service Provider (SMSP) to undertake a comprehensive, long-term study on eSIM-enabled mobile IoT. This collaboration, which began in 2020, involves managing eSIMs for over 60 million IoT devices globally. Specifically in China, there are 22,208,536 registered IoT devices, of which 15,063,363 remain active as of September 2024 [1]. Our extensive analysis draws from over 11 billion activity records of these devices, collected from 2021 to 2024. These records provide a detailed log of critical events such as registration, deregistration, operator switch requests, and connectivity failures. Below, we summarize the principal findings from this substantial dataset:

- *Business Statistics.* The average lifespan of IoT devices is approximately 2.35 years, although this varies significantly across different industries, ranging from 12 to 36 months. About 75% of these devices consume less than 100 MB of data monthly. The remaining 25% of devices, which are considered premium users, require more than 100 MB and up to 1 GB of bandwidth, with 93% of these

[1]The number of registered devices equals the total devices minus those deregistered, while an active device sends at least one request per month.



(a) The Existing Failover based MOA (FOMOA)



(b) The Proposed Lookahead based MOA (LAMOA)

Figure 3.1: Illustration of Two Multi-Operator Access (MOA) Strategies. (a) The device is using the local operator A's network for business data transmission. When moving to P2, it loses Internet connectivity and establishes a temporary connection via a universal profile, which involves costly roaming through the registered operator X. This temporary connection is used to fetch a new profile from our platform, enabling the device to reconnect to the local operator B. This MOA approach requires two time-consuming operator switches. (b) Before the device exits the coverage area of the current operator, the platform anticipates this by proactively loading a new profile onto the device. When the device disconnects, it automatically installs this profile to seamlessly connect with local operator B, with a single operator switch.

devices facing substantial mobility challenges.

- *Performance Statistics.* Our data shows that 97% of sampled locations in China benefit from coverage by at least two mobile operators, creating a strong foundation for a multi-operator access environment. Devices equipped with eSIM on our platform, on average, initiate operator switches 15 times daily, with 90% of these devices making up to 20 switch requests each day. While switching to a new operator's network takes only 10 seconds, the current multi-operator access (MOA) strategy results in a handover latency of up to 10 minutes.

Our study affirms the critical role and growing need for eSIM-enabled mobile IoT, as well as spotlights the existing shortcomings in the current failover based multi-operator access (FOMOA) strategy employed in the industry. As illustrated in Fig. 3.1(a), FOMOA triggers a switch only when an IoT device loses connectivity, relying on an expensive universal profile as a bridge to secure a new local profile (more details refer to §3.3). Although this method is straightforward and effective, it is highly inefficient in operational terms, causing an average business interruption of 10 minutes.

In this work, we introduce *lookahead multi-operator access* (LAMOA) strategy, which utilizes advanced large AI models to accurately predict future operator switch events. This method allows our platform to proactively load the necessary eSIM profiles and direct devices to switch to alternate networks ahead of potential disconnections, as depicted in Fig. 3.1(b). This anticipatory approach minimizes the risk of disconnection and network service interruptions while significantly reducing the time and costs associated with switching. Moreover, LAMOA enhances eSIM management by centralizing decision-making for switch timing and operator selection, thereby improving operational efficiency and reducing the computational burden on devices. However, the deployment of LAMOA faces two significant challenges:

• *How to determine switch needs without priors?* Accurate spatial information about

devices is unavailable due to the need to safeguard customer confidentiality. Additionally, channel information is inaccessible because channel scanning is energy-intensive for IoT devices and disrupts data transmission. The absence of spatial and channel priors makes it challenging to predict switch timings accurately. However, our platform has collected billions of switch requests from millions of devices over the past years, creating a robust foundation for identifying consistent behavioral patterns across devices over time. This extensive dataset effectively addresses the forecasting challenge posed by the lack of spatial and channel priors.

• *How to develop the forecast model?* Our platform manages over 1.5 million switch transactions each month, with an average of approximately 50,000 switches per hour. The sheer volume and high frequency of these events pose significant challenges for accurate forecasting. To address this, we designed and implemented a forecasting framework, termed NeSIM [2] for the LAMOA. We leverage a pre-trained time-series forecast model as the backbone to extract multi-scale features from historical data spanning the last 24 hours, week, month, and year. These features capture underlying patterns at various levels of granularity. We then utilize a Transformer decoder with learnable queries to segment events from these multi-scale features. This approach not only harnesses the strengths of the large time-series model but also effectively addresses challenges related to temporal event density.

Contribution. This study is pioneering in its examination of eSIM technology's performance on a large scale. Our data showcases eSIM's effectiveness in significantly enhancing the scalability of mobile IoT, marking a substantial advancement in the field. Our comprehensive analysis sheds light on various critical aspects such as device lifespan, bandwidth requirements, operator coverage, switching frequency, and the costs associated with switching. Notably, we identify that the prevalent industrial switching strategy is inadequate, highlighting the need for more efficient approaches to network management in IoT environments. To this end, our work is the first to

[2] NeSIM embodies the neural eSIM technology.

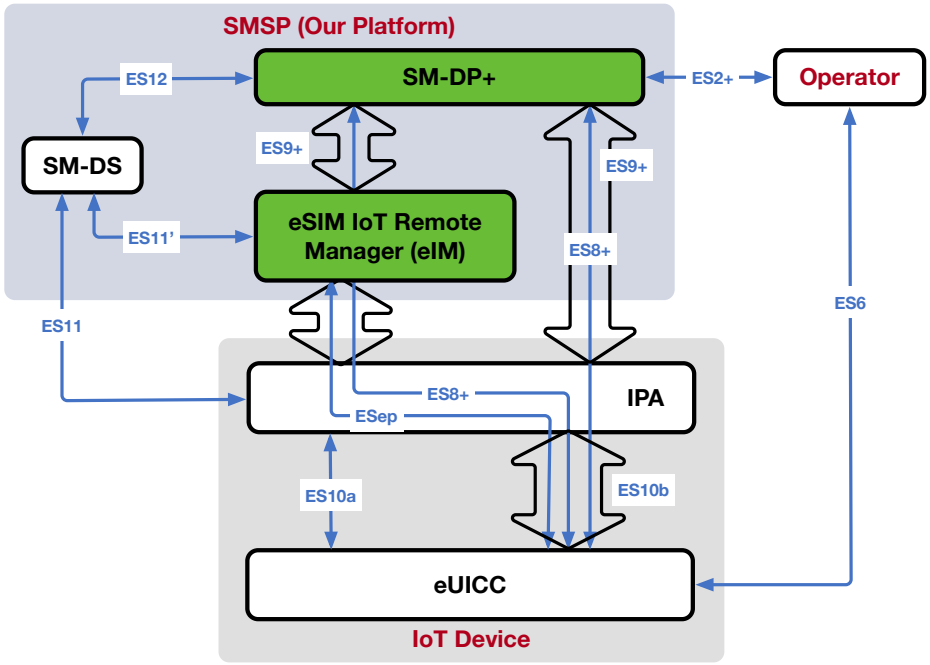


Figure 3.2: Functional Architecture of eSIM enabled IoT

design and implement NeSIM, the efficacy of which has been rigorously validated and tested.

3.1 Primers

This section offers a comprehensive analysis of the technical architecture and functionality of the eSIM-enabled mobile IoT system.

3.1.1 Architecture of eSIM-enabled IoT

The architecture of eSIM-enabled IoT is depicted in Fig. 3.2, which adheres to the GSMA technical specification [46]. This framework comprises three primary roles —Operator, SMSP (Subscription Management Service Provider), and IoT Device— along with five core components: SM-DP+ (Subscription Manager - Data Preparation +), SM-DS (Subscription Manager - Discovery Server), IPA (IoT Profile Assistant),

eIM (eSIM IoT Remote Manager), and eUICC (embedded Universal Integrated Circuit Card). These components are interconnected via well-defined logical interfaces, such as ES9+, ES8+, ES5 and HTTPS/TLS, enabling seamless remote provisioning and management of eSIM profiles.

• **Role 1: Operator**: Typically a mobile network operator (MNO) or a mobile virtual network operator (MVNO), the operator provides wireless network services and issues eSIM profiles that enable customers to connect to its network.

• **Role 2: SMSP**: The SMSP is an entity that manages eSIM profiles and service subscriptions for IoT devices, facilitating the remote provisioning and management of eSIM profiles directly onto devices. It contains three key components. (1) SM-DP+: This component is essential for generating, downloading, and remotely managing eSIM profiles. Typically under the ownership and maintenance of the operator, the SM-DP+ handles secure requests and facilitates the delivery of protected profile packages to IoT devices. (2) SM-DS: Acting as a bridge, it allows the SM-DP+ to send notifications or post events to the SM-DS, which are then pulled by the device's LPAs as required. (3) eIM: The eIM plays a crucial role in orchestrating Profile State Management Operations (PSMO) for IoT devices. By interfacing with the SM-DP+ and SM-DS, the eIM functions as a proxy to enhance data exchanges and enable IoT devices to overcome user interface and network connectivity constraints.

• **Role 3: IoT Device**. An IoT device is equipped with two main components: (1) IPA: This software component facilitates eUICC provisioning and is tailored to meet the specific needs of IoT environments. It supports crucial functions such as Discovery Service, Profile Download, Profile State Management Operation Conveying, and Notification Handling, which are essential for managing eSIM profiles efficiently. (2) eUICC: This is a secure, tamper-resistant hardware element that stores eSIM profiles and cryptographic certificates. Although it can hold multiple profiles, GSMA standards generally limit the eUICC to having one active profile at a time.

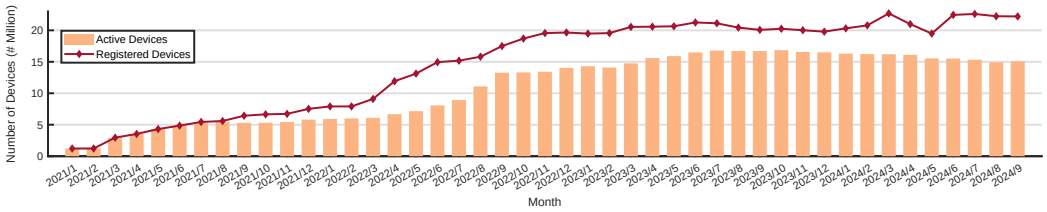


Figure 3.3: Cumulative Number of Registered Devices and Active eSIM-Enabled IoT Devices on our SMSP Platform (2021–2024)

3.1.2 eUICC

The eUICC is the physical component embedded within IoT devices, capable of hosting one or more eSIM profiles. An eSIM profile contains all the essential configuration data required to define a subscriber's mobile service, including the subscriber's identification number, operator credentials, and access keys necessary for authentication and establishing network connections. Specifically, we employ a dual-profile eUICC design, managing two distinct profiles: a universal profile and a business profile.

- **Universal Profile**: This profile provides connectivity to any operator in emergency situations. Typically issued by a non-local operator, it often necessitates remote roaming across various operators' networks, thereby incurring higher operational costs. For management purposes, we pre-install a universal profile on each device.
- **Business Profile**: Issued over-the-air by our platform, this profile is obtained from a local operator tailored to the client's business locale, which helps to minimize service costs. It is specifically utilized for transmitting business-related data for our clients.

When initially activated or upon losing connectivity, the device utilizes the universal profile to acquire and install a local operator's profile (business profile) for business operations. Subsequently, the device switches to the local operator using the installed business profile. This dual-profile design ensures consistent and reliable connectivity, effectively balancing cost efficiency with broad network accessibility.

3.1.3 Remote SIM Provisioning via SMSP

Remote SIM Provisioning (RSP) enables the remote activation, switching, and deactivation of SIM profiles over-the-air. Our partner matains a vast repository of eSIM profiles from various operators. This allows any registered device to seamlessly obtain profiles, ensuring flexibility and accessibility. The process begins with our platform securing and storing profiles in a pool, pre-equipping devices with a default SM-DP+ address and universal profile for initial activation. When a device loses connectivity, it temporarily connects via the universal profile to request new profiles. Our platform then compiles and sends a profile package to the device, which sequentially tests each profile until establishing a connection. Once a profile is activated, the device reports the outcome to the platform and resumes data transmission. Additionally, our platform supports active network switching, where the platform initiates the process instead of the device.

3.2 Measurement Results

In this section, we first provide an overview of our eSIM platform, followed by an analysis of business and perform ance statistics derived from the logged data.

3.2.1 General Statistics

Over the last four years (2021-2024), our platform successfully issued nearly 60 million eUICCs. Fig. 3.3 presents the cumulative number of registered and active devices from 2021 to 2024, associated with our eSIM service. Active devices are defined as those that have sent at least one request within the month. The monthly number of registered devices includes all previously registered devices plus any new registrations, while also subtracting devices that have been deregistered due to expiration

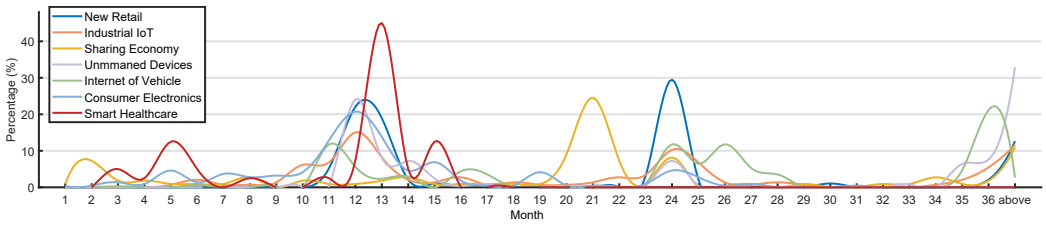


Figure 3.4: Lifespan of IoT Devices across Various Industrial Sectors

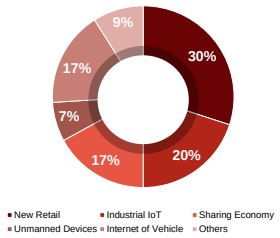


Figure 3.5: Sectoral Distribution

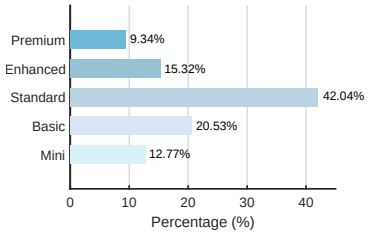


Figure 3.6: Monthly Data Plan

or non-use. By the end of September 2024, the data indicates a total of 22,208,536 devices registered, of which 15,063,363 remain actively engaged with the service. This engagement rate implies that, on average, approximately 78% of registered devices actively utilize the service monthly. The graph demonstrates the sustained growth in device registrations and the somewhat stable proportion of those that remain active.

In addition, we measured the lifespan of IoT devices, defined as the period from when a device is registered on our platform to when it is deregistered. Fig. 3.4 illustrates the lifespan of IoT devices across various industrial sectors. Specifically, the peak lifespans for devices in new retail, industrial IoT, sharing economy, unmanned devices, internet of vehicles, consumer electronics, and smart healthcare are 24, 12, 21, 36, 36, 12, and 13 months, respectively. The average lifespan is 22 months, shorter than the 28-month lifespan of high-end cellular devices such as smartphones or wearables [47].

These visual representations provide a clear insight into the growth of eSIM adoption, showcasing its significance in enabling the scalability of mobile IoT in real-world scenarios.

3.2.2 Business Statistics

Next, we compiled data from a business perspective.

(1) Utilization Distribution. Fig. 3.5 illustrates the distribution of eSIM-enabled IoT usage across various industrial sectors. The New Retail sector, leveraging eSIMs for POS terminals and logistics, leads with 30% of total usage. Industrial IoT follows at 20%, primarily enhancing machine-to-machine communications and manufacturing efficiencies. Both the Sharing Economy and Unmanned Device sectors each account for 17%, with the former supporting services like car and bike sharing, and the latter encompassing drones and robotics. The Internet of Vehicles, which improves navigation and vehicle diagnostics, represents 9%, while miscellaneous applications, such as smart gas meters, make up the remaining 7%. Notably, 93% of these applications face mobility challenges and benefit from seamless network switching and remote management from eSIM technology.

(2) Service Plan. We offer five tiered data plans to meet our clients' varying needs: Mini (0-5MB), Basic (5-30MB), Standard (30-100MB), Enhanced (100MB-1GB), and Premium (over 1GB). Each plan is designed to cater to different levels of data usage, enabling customers to choose options that best fit their operational needs and budget constraints. Fig. 3.6 displays the distribution of these plans across devices. The Mini plan, chosen by 12.77% of devices, is ideal for minimal data uses like occasional connectivity in smart meters. The Basic plan supports basic tracking or monitoring with 20.53% of devices using it, while the Standard plan, which accommodates moderate usage, is preferred by 42.04% of devices. These two plans are often used for the sharing economy and new retail sectors. The Enhanced plan, selected by 15.32% of devices, and the Premium plan, chosen by 9.34% of devices. These two plans serve for the applications with substantial and high data demands (e.g., surveillance video streaming). This structure underscores that our platform covers a broad range of data requirements, from the very minimal to the

most intensive.

3.2.3 Performance Statistics

While our partner operates across 20 countries or regions worldwide, our primary focus is on China. Therefore, we have selected service data from China in the past four years to conduct a detailed performance analysis.

(1) Operator Coverage. In China, cellular services are predominantly provided by three major operators: China Mobile, China Unicom, and China Telecom. We are collaborating with these operators to ensure a reliable supply of eSIM profiles and extensive national coverage. According to an official report [48], the country boasts over 10 million cellular base stations, including 3.38 million 5G stations and more than 6 million 4G stations. For compatibility, our IoT devices primarily use the LTE or LTE-M air interface standards. To evaluate operator coverage, we instructed 1,000 IoT devices to report their current locations and scan for available operators at the locations. The results, illustrated in Fig. 3.7, show that 2.3% of locations are served by a single operator, 65.53% by two operators, and 32.17% by all three operators. This extensive network coverage underscores the widespread availability of operator options nationwide.

(2) Switching Frequency. We analyzed the switching frequency across devices that requested at least two operator switches, measured as the number of daily switch requests per device. On average, devices underwent 15 network switches per day, with 90% of devices executing more than 20 switches and a maximum of 650 switches in extreme cases. The standard deviation was significantly high at 55 switches per day. Higher daily switching frequencies were primarily observed in high-mobility sectors, such as the sharing economy and the Internet of Vehicles. These findings highlight the essential role of frequent operator switching in accommodating the dynamic needs of mobile IoT environments.

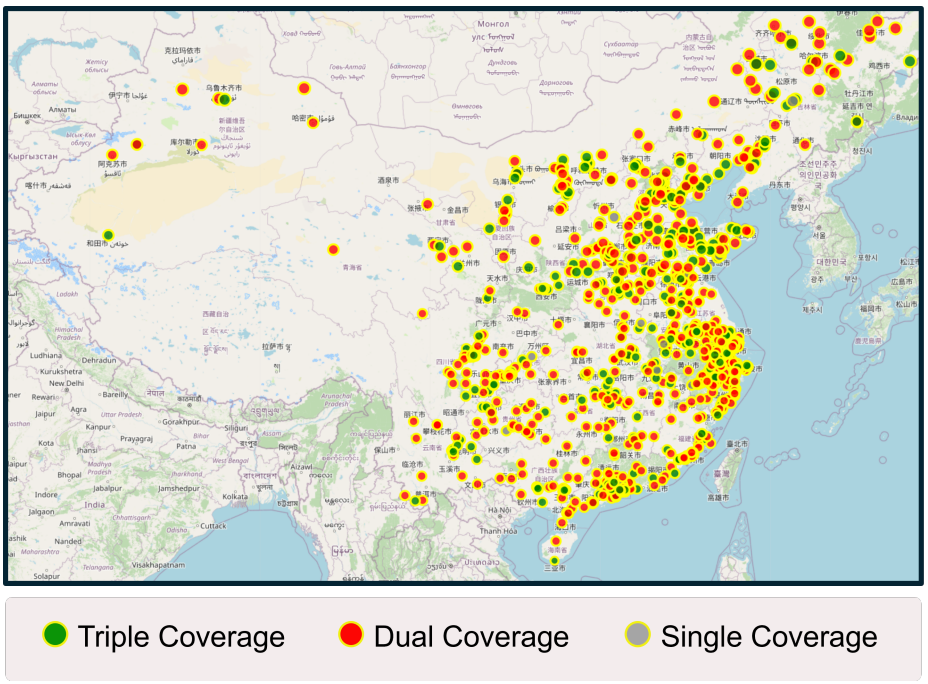


Figure 3.7: Operator Coverage

(3) Switching Time. We assessed the time required for operator switching per request through two methods. First, we calculated the time from the initiation of a switch request to the successful completion of the switch by the platform. Second, we performed controlled experiments by issuing AT commands to testbed devices as a baseline. The results are depicted in Fig. 3.8, labeled as "Logged" for the operational logs and "Tested" for the testbed results. The testbed method showed an average switching time of 10.42 seconds (median: 9.83, P_{90}: 13.13), whereas the log-based method averaged at 11.7 seconds (median: 10.56, P_{90}: 13.92). The slight increase of 1.3 s may typically be attributed to database interaction delays. Thus, logged data closely mirrors actual conditions, making it suitable for both analysis and predictive modeling.

(4) FOMOA Latency. We measured the latency of the FOMOA strategy by tracking the time elapsed from a device's disconnection to its reconnection to the business server. The results are shown in Fig. 3.9. The complete FOMOA process averages a latency of 10.06 minutes (median: 8.99; P_{90}: 17.76). This latency is nearly 60×

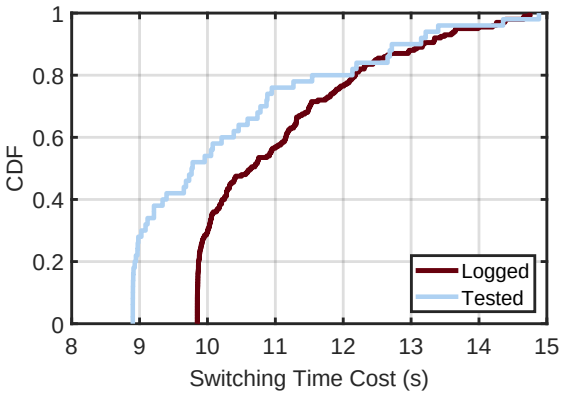


Figure 3.8: Switching Time

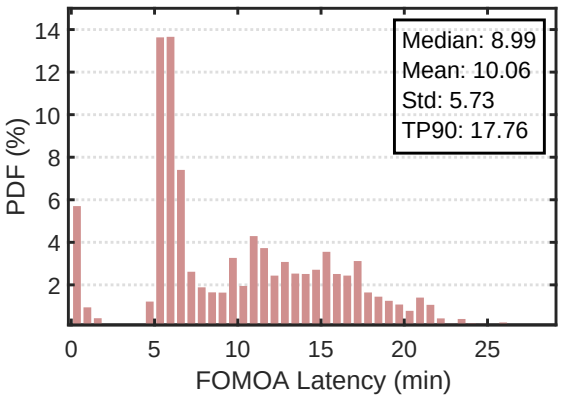


Figure 3.9: MOA Latency

longer than the actual time required to switch from one operator to another. This extended latency is primarily attributed to several factors. First, a prolonged timeout mechanism (approximately 1-2 minutes) is used to detect disconnections, which helps prevent frequent, unnecessary switches caused by transient network fluctuations—a phenomenon known as the ping-pong effect. Second, the FOMOA strategy requires two distinct operator switches: the first switch accesses an available operator network using the universal profile, and the second switch connects to a local operator to resume business activities. Notably, each switch involves a time-intensive channel scanning process (approximately 3-4 minutes) to identify an available operator network, significantly contributing to the overall latency.

3.2.4 Summary

Our successful four-year, million-scale business service experience enables us to fully realize the potential of eSIM technology in scaling mobile IoT deployments in real-world scenarios. The extensive operator coverage and frequent operator switch activities observed further underscore the critical need for such capabilities. However, the current industry-standard FOMOA strategy is static and relatively rudimentary. There is an urgent need to develop and refine a new MOA strategy that can more effectively address the dynamic demands of large-scale mobile IoT networks.

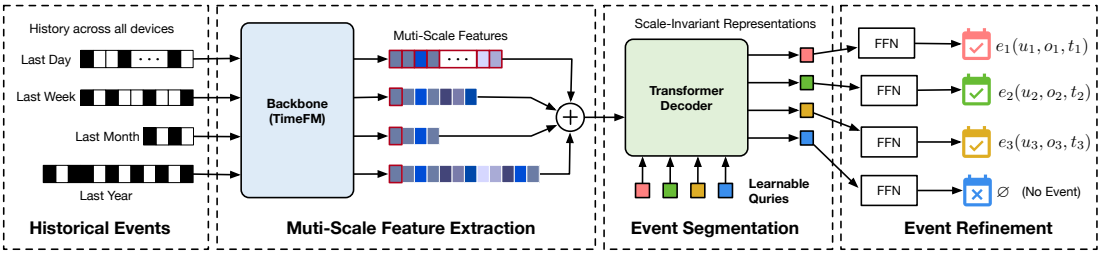


Figure 3.10: Architecture of NeSIM. The system utilizes a pre-trained large time-series model as the backbone to extract multi-scale features. These features are combined and subsequently processed by a Transformer Decoder equipped with learnable queries. Each output token is passed through a FFN to predict events for the following day.

3.3 System Design

This section presents a data-driven MOA strategy designed to address the latency challenge inherent in the FOMOA.

3.3.1 Motivation

Although switching from one operator to another typically takes only 10 seconds, FOMOA often requires up to 10 minutes to transfer an IoT device to a different operator after disconnection, due to delays from timeouts and channel scanning. This significant gap has driven us to develop the Lookahead MOA (LAMOA) strategy, which aims to minimize latency. As Fig. 3.1(b) shows, LAMOA proactively forecasts when a switch is necessary and initiates the transition to the next operator's network using the existing local connection before internet access is compromised. By eliminating the need for intermediate steps such as waiting for timeouts, searching for available operators, establishing temporary connections via universal profiels, and executing additional operator switches, LAMOA substantially reduces latency.

Problem Formulation. At the heart of the LAMOA is its predictive capability: forecasting when and which devices will lose network connectivity and identifying the optimal operator for the switch. Formally, each network switch event e_i is character-

ized by a tuple (u_i, o_i, t_i), where device $u_i \in \mathcal{U}$ switches to operator $o_i \in \mathcal{O}$ at time t_i. For each device u, we establish an intensity function $f_u(t, o)$, which quantifies the immediate likelihood of the device switching to operator o at time t. Using historical event data $\{e_1, e_2, \ldots, e_{i-1}\}$ for device u, the forecast for the next switch time $\hat{t}_i$ is calculated using the following integral:

$$\hat{t}_i = \int_{t_{i-1}}^{\infty} t f_u(t) \exp\left(- \int_{t_{i-1}}^{t} f_u(s) ds \right) dt \tag{3.1}$$

where

$$f_u(t) = \sum_{o \in \mathcal{O}} f_u(t, o) \tag{3.2}$$

The $f_u(t)$ sums the intensity across all possible operators.

3.3.2 High-Level Design

Accurately fitting the intensity function falls within the domain of deep learning. To this end, we have developed a sophisticated event-series forecasting framework, NeSIM, tailored to address the unique demands of network switching forecasts. As shown in Fig. 3.10, NeSIM is structured around three core components:

- **Backbone Encoding**: NeSIM leverages a time-series model as the backbone to extract multi-scale features from switch events across all devices. These features are derived from event data spanning the last day, last week, last month, and last year.
- **Event Segmentation**: This component segments the upcoming 24-hour switch events from the multi-scale features using a deformable Transformer Encoder with learnable queries. This method enables effective cross-attention, allowing NeSIM to discern complex temporal patterns across various granularities.
- **Event Refinement**: Multiple Feed-Forward Networks (FFNs) refine the details of each segmented event, by classifying devices and operators and identifying the time point using the linear regression.

The subsequent sections will delve deeper into each component, exploring their specific roles and how they interact within the overarching framework.

3.3.3 Backbone Encoding

We utilize TimeFM [49] as the primary backbone encoder for our forecasting framework [3]. TimeFM employs a Transformer-Decoder architecture tailored to process a wide array of time-series data, making it highly adaptable to various forecasting scenarios. With 200 million parameters, TimeFM has been extensively pre-trained on a dataset that includes 100 billion data points drawn from Google Trends, Wikipedia page view statistics, and synthetic time series.

Formally, the operational mechanism of TimeFM can be described by:

$$\langle e_{n+1}, e_{n+2}, \ldots, e_{2n} \rangle = f_{\text{TimeFM}}(\langle e_1, e_2, \ldots, e_n \rangle) \tag{3.3}$$

where both the input and output are sequences of time-series data at consistent intervals. Our objective is to leverage historical data ranging from a day to a week or even a month to predict network switches in the upcoming 24 hours. However, TimeFM's capabilities do not entirely meet our specific needs. Firstly, due to its design, TimeFM predicts the next n time points solely based on the preceding n time points, which means a 24-hour forecast is dependent only on the previous 24 hours of data. This limitation overlooks longer historical contexts, which are crucial for the devices with infrequently switch requests. Secondly, while TimeFM has the capability to extend its forecasting horizon through a patching method that aggregates minute-scale events into larger temporal blocks, finding the ideal patch size to accommodate the diverse switching patterns of millions of devices is challenging. Lastly, because TimeFM treats the event series of each device as independent, executing forecasts for millions of devices individually significantly increases the computational burden.

[3]Other time-series models, such as PatchTST [50], can also be utilized as the backbone encoder.

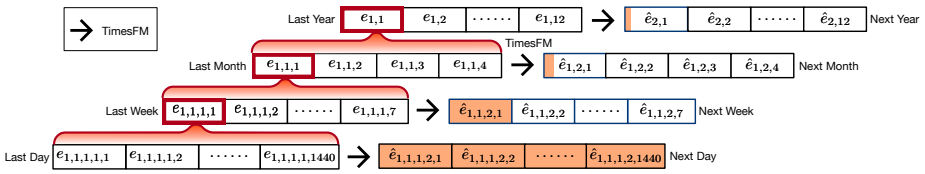


Figure 3.11: Multi-scale Feature Extraction

Multi-Scale Feature Extraction: To address the above challenges, we have implemented a multi-scale feature encoding method. We organize switch events at a minute-scale, each represented as $e_{y,m,w,d,i}$, where y, m, w, d and i correspond to the year, month, week, day, and minute of occurrence, respectively. For simplicity, higher dimensional representations are used for a vector of events with lower dimension as follows.

$$\begin{cases} e_{y,m,w,d} = \langle e_{y,m,w,d,1}, e_{y,m,w,d,2}, \ldots, e_{y,m,w,d,1440} \rangle \\ e_{y,m,w} = \langle e_{y,m,w,1}, e_{y,m,w,2}, \ldots, e_{y,m,w,7} \rangle \\ e_{y,m} = \langle e_{y,m,1}, e_{y,m,2}, e_{y,m,3}, e_{y,m,4} \rangle \end{cases} \tag{3.4}$$

The above equations indicate that each day consists of 1440 minute-scale events, each week comprises 7 day-scale events, and each month includes 4 week-scale events. Specifically, $e_{y,m,w,d,i} \in \mathbb{R}^{1\times 3}$, $e_{y,m,w,d} \in \mathbb{R}^{1,440\times 3}$, $e_{y,m,w} \in \mathbb{R}^{10,080\times 3}$, and $e_{y,m} \in \mathbb{R}^{43,200\times 3}$. Each minute-scale event contains the device ID, time point and the operator. In the event series, an event entry might be empty, indicating that no event occurred at that minute. As depicted in Fig. 3.11, we patch minute-scale switch events into four distinct scales: daily, weekly, and monthly. These event series are sequentially processed by four TimeFM models respectively .

- *Daily Forecast*: Treating each minute-scale event as a separate token, we input 1,440 events from the previous day into TimeFM to forecast the events for the upcoming day.

$$\langle \hat{e}_{1,1,1,2,1}, \ldots, \hat{e}_{1,1,1,2,1440} \rangle = f_{\text{TimeFM}}(\{e_{1,1,1,1,1}, \ldots, e_{1,1,1,1,1440}\})$$

This directly provides 1,440 minute-scale event forecasts.

- *Weekly Forecast*: Patching each day's 1,440 minute-scale events into a single day-scale token, we input the last seven day-scale patches into TimeFM to forecast the events for the upcoming week.

$$\langle \hat{e}_{1,1,2,1}, \dots \hat{e}_{1,1,2,7} \rangle = f_{\text{TimeFM}}(\langle e_{1,1,1,1}, e_{1,1,1,2}, \dots, e_{1,1,7} \rangle)$$

  This yields 7 day-scale event forecasts, with only the first day's forecast being directly useful for our purposes.
- *Monthly Forecast*: We aggregate the 10,080 events from a week into a single week-scale patch and use four such patches to forecast the events for the upcoming month.

$$\langle \hat{e}_{1,2,1}, \hat{e}_{1,2,2}, \dots, \hat{e}_{1,2,4} \rangle = f_{\text{TimeFM}}(\langle e_{1,1,1}, e_{1,1,2}, \dots, e_{1,1,4} \rangle)$$

  This results in four week-scale event forecasts.
- *Yearly Forecast*: Each month's 43,200 events are condensed into a single month-scale patch, with twelve such patches used to predict events for the upcoming year.

$$\langle \hat{e}_{2,1}, \dots, \hat{e}_{2,12} \rangle = f_{\text{TimeFM}}(\langle e_{1,1}, e_{1,2}, \dots, e_{1,12} \rangle)$$

  This provides 12 month-scale event forecasts.

Given that the lifespan of most IoT devices is under two years (as discussed in §3.2), we typically utilize data from the last one year at most. Here, we adopt natural time scales (i.e., day, week, and month) to aggregate events, as we believe the activities of IoT devices are inherently aligned with human work cycles. Recall that our focus is on forecasting events occurring within the next 24 hours. Therefore, we extract the 1440×4 minute-scale forecast results relevant to the next 24 hours (highlighted in red) to construct a multi-scale feature vector $\mathcal{F}$ for imminent predictions as follows:

$$\mathcal{F} = \underbrace{\hat{e}_{2,1}[1:1440]}_{\text{Yearly}} + \overbrace{\hat{e}_{1,2,1}[1:1440]}^{\text{Monthly}} + \underbrace{\hat{e}_{1,1,2,1}}_{\text{Weekly}} + \overbrace{\langle \hat{e}_{1,1,1,2,1}, \hat{e}_{1,1,1,2,2}, \dots, \hat{e}_{1,1,1,2,1440} \rangle}^{\text{Daily}}$$

In this way, we aim for TimeFM to uncover deeper, long-term patterns by leveraging more extensive historical data, while still aligning with TimeFM's operational framework.

3.3.4 Event Segmentation and Refinement

Inspired by previous segmentation technique [51], we utilize a standard Transformer Decoder to segment switch events from the multi-scale features processed by TimeFM. Illustrated in Fig. 3.10, NeSIM creates a fixed-sized set of N_Q event forecasts, setting N_Q significantly higher than the usual number of events within 24 hours. We establish N_Q learnable queries, referred to as Q, and the decoder computes cross-attention with the input feature $\mathcal{F}$ using the following formula:

$$\text{Atten}(Q, \mathcal{F}, \mathcal{F}) = \text{Softmax}\left(\frac{QW_Q \cdot \mathcal{F}W_K^T}{\sqrt{d_k}}\right)\mathcal{F}W_V \tag{3.5}$$

where d_k represents the dimensionality of the keys, while W_Q, W_K, and W_V are the weight matrices for the queries, keys, and values, respectively.

With N_Q queries inputted, the decoder outputs N_Q tokens, each potentially representing an event. A Feed-Forward Network (FFN) is used to extract event details, including device ID, time point in minute-scale, and the target operator ID. The device ID and operator ID are classified, while the timepoint is determined through linear regression. Given the typical number of actual events is less than N_Q, tokens not corresponding to actual events will denote the device ID as $\varnothing$, which are then directly ignored.

This approach offers two significant advantages. Firstly, it enables the processing of events from all devices in a single batch, which effectively reduces computational costs. Secondly, it effectively manages simultaneous events, a scenario that TimesFM is not equipped to handle.

3.3.5 Loss Function

After filtering out empty events, our framework predicts $\hat{k} \leq N_Q$ events, represented as:

$$\hat{E} = \{\hat{e}_1(\hat{u}_1, \hat{t}_1, \hat{o}_1), \hat{e}_2(\hat{u}_2, \hat{t}_2, \hat{o}_2), \ldots, \hat{e}_{\hat{k}}(\hat{u}_{\hat{k}}, \hat{t}_{\hat{k}}, \hat{o}_{\hat{k}})\}$$

Meanwhile, the ground truth, i.e., the actual switch events occurring within the next 24 hours, consists of k events:

$$E = \{e_1(u_1, t_1, o_1), e_2(u_2, t_2, o_2), \ldots, e_k(u_k, t_k, o_k)\}$$

The two sets are not inherently ordered, and their sizes may differ. Our training objective is to minimize the discrepancy between these sets. To achieve this, we design a joint loss function comprising device loss and event loss.

Device Loss. Accurate device classification is paramount, as incorrect predictions lead to unnecessary switching in other devices and fail to address the correct device's needs. Thus, we define the device loss using the Jaccard index to measure the accuracy of device predictions:

$$\mathcal{L}_d = 1 - \frac{|U \cap \widehat{U}|}{|U \cup \widehat{U}|}$$

where $U = \{u_1, u_2, \ldots, u_k\}$ represents the true devices and $\widehat{U} = \{\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_{\hat{k}}\}$ represents the predicted devices. Note that duplicates should be excluded when calculating the Intersection over Union (IoU).

Event Loss. For the M devices common to both U and $\widehat{U}$, we separate and order their events by time:

$$\begin{cases} \widehat{E}(u_m) = \{\big(\hat{t}_1(u_m), \hat{o}_1(u_m)\big), \big(\hat{t}_2(u_m), \hat{o}_2(u_m)\big), \ldots\} \\ E(u_m) = \{\big(t_1(u_m), o_1(u_m)\big), \big(t_2(u_m), o_2(u_m)\big), \ldots\} \end{cases}$$

where $m = 1, 2, \ldots, M$. Here, $\widehat{E}(u_m)$ and $E(u_m)$ denote the predicted and ground truth event-series for device $u_m \in U \cap \widehat{U}$, respectively. To ensure temporal alignment

between these series, we employ the Dynamic Time Warping (DTW) distance as the event loss:

$$\mathcal{L}_e = \frac{1}{M} \sum_{m=1}^{M} \mathrm{DTW}(\widehat{E}(u_m), E(u_m))$$

Unlike Euclidean distance, DTW accommodates non-linear temporal distortions, making it ideal for comparing time series with varying alignments.

Joint Loss. The overall loss function combines the device loss and event loss:

$$\mathcal{L} = \lambda \mathcal{L}_d + (1 - \lambda)\mathcal{L}_e$$

where λ is a hyper-parameter balancing the two components.

3.3.6 Proactive Scheduling

At the start of each day, we conduct a forecasting session and subsequently develop a schedule that proactively instructs IoT devices to switch to alternative operators in advance. This preemptive strategy is designed to optimize network efficiency and ensure seamless connectivity for all devices throughout the day. We also maintain the FOMOA as a fallback option to safeguard against any inaccuracies in our predictions. Should the prediction fail, the IoT device is programmed to revert to our existing passive approach, which involves requesting the business profile when it detects a disconnection from the current network.

3.3.7 Implementaion

Configuration. We have integrated TimeFM v2 [52] as our core backbone encoder. This model is capable of forecasting up to 2049 timepoints, significantly exceeding our requirement of 1440 timepoints for daily minute-scale forecasts. The event segmentation is implemented using a classical Transformer decoder. The decoder consists of

Table 3.1: Summary of Forecast Performance

Model	Device Forecast				Operator Forecast				Overall		
	Precision	Recall	F1	IoU	Precision	Recall	F1	IoU	Precision	Recall	F1
PatchTST	75.62%	69.01%	0.752	80.67%	72.2%	73.6%	0.729	79.32%	54.59%	50.79%	0.526
TimesFM	76.13%	73.8%	0.749	83.45%	79.4%	75.18%	0.772	81.65%	60.4%	55.48%	0.578
PatchTST+Seg.	83.5%	76.16%	0.797	89.28%	89.33%	78.71%	0.837	92.79%	74.6%	59.95%	0.665
TimesFM+Seg. (NeSIM)	**86.9%**	**81.67%**	**0.842**	**92.84%**	**92.1%**	**85.27%**	**0.886**	**94.12%**	**80.03%**	**69.64%**	**0.745**

6 layers, each with 8 attention heads, and operates with a model dimension of 256. The decoder's architecture comprises approximately 20 million parameters. We adopt $N_Q = 10,000$ learnable queries and $\lambda = 0.7$. The final event refinement is performed by a 3-layer MLP with a ReLU activation function.

Dataset. Due to hardware limitations, our dataset was limited to 30,000 active devices, which are deployed across 21 provinces, 4 municipalities, and 5 autonomous regions. Given the 2.3-year average lifespan of IoT devices, our dataset only comprises 16,953,796 request event logs collected from these devices over the past two years. Samples of these logs document the timing of switch requests, the devices involved, and the operator networks they transitioned from and to. We utilized the first 19 months of data (80% of the dataset) for training and allocated the remaining 5 months (20% of the dataset) for testing.

Training. Since TimesFM is well pre-trained, we keep its parameters fixed during the training phase, focusing our efforts on optimizing the segmentation decoder to better align with our operational requirements. Our training regimen spans 10 epochs, with the learning rate reduced by a factor of 10 after the first 5 epochs. Each epoch involves processing all training data once. On a single A100 GPU, training the model for 10 epochs requires approximately 48 hours.

3.4 Evaluation

This section evaluates the advancement of NeSIM and the performance of various fine-tuning techniques.

3.4.1 Forecast Performance

We commence our evaluation by assessing the forecast performance for the classification tasks within NeSIM.

Baseline. To facilitate a comprehensive comparative analysis, we also implement LAMOA using PatchTST [50], a state-of-the-art Transformer-based time-series forecasting model, as the backbone feature extractor. This configuration allows us to explore four distinct methods: (1) solely utilizing PatchTST, (2) solely utilizing TimesFM, (3) combining PatchTST with our proposed segmentation decoder, and (4) combining TimesFM with our proposed segmentation decoder (NeSIM). The first three methods serve as baselines for comparison against NeSIM.

Results. Both device and operator forecasts are performed using classification techniques. Accordingly, we employ standard evaluation metrics such as precision, recall, F1 score, and IoU to measure the performance of our models. The evaluation results are shown in Table 3.1. The key findings are summarized as follows:

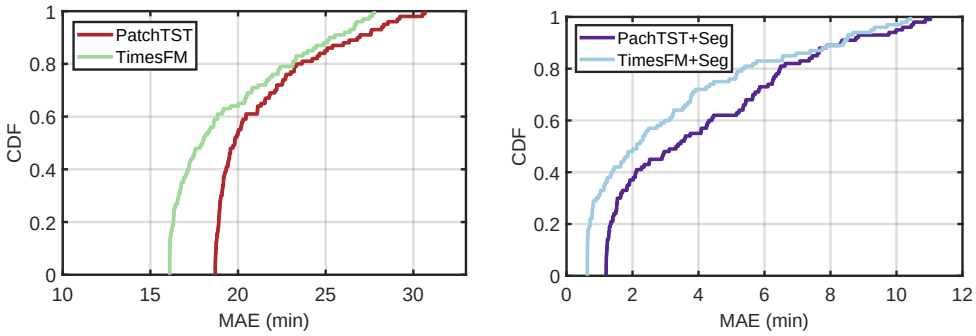
- **Device Forecast**: We compare the sets of predicted devices and the ground truth using the four metrics. As a result, the standard time-series models (PatchTST or TimesFM) achieve similar results, with averages of 76% precision, 71% recall, 0.75 F1 score, and 82% IoU. In contrast, the enhanced model incorporating a segmentation decoder demonstrates superior performance, achieving averages of 83% precision, 80% recall, 0.82 F1 score, and 91% IoU. Notably, NeSIM outperforms PatchTST + Segmentation, with improvements of 3.4% in precision, 5.51% in recall, 4.5% in F1 score, and 3.56% in IoU.
- **Operator Forecast.** The forecast results are categorized by device into switch event series, which are organized chronologically. We then utilize DTW to align the predicted event series with the ground truth. Subsequently, we calculate the four metrics on the aligned event series regarding the operator. The PathTST and TimesFM achieve average metrics of 75.80% precision, 74.39% recall, 75.05%

F1 score, and 80.49% IoU. NeSIM surpasses these standard models, improving by averages of 16.3%, 10.88%, and 13.55% across the four metrics. Additionally, NeSIM outperforms the PatchTST+Segmentation solution by 2.7%, 6.56%, and 4.9% respectively.

- **Overall Forecast**. As previously noted, the performance of the operator forecast hinges on the correct classification of devices. Precision and recall in this context are understood as conditional probabilities, *i.e.*, $P(O|D)$. To calculate the overall forecast precision and recall, we use the formula $P(O) = P(D)P(O|D)$. Consequently, our proposed model achieves a precision of 80.3% and a recall of 69.64%. These figures represent improvements over the standard time-series models by 23% and 16.5% for precision and recall, respectively. Moreover, XiTuXi surpass the performance of the PatchTST+Segmentation solution by 5.43% in precision and 9.69% in recall.

Summary. Notably, utilizing standard time-series models (PatchTST or TimesFM) directly to address our problem results in a notable poor performance compared to our enhanced solution, which combines these models as backbones with segmentation decoders. This underperformance is primarily due to two factors. First, the intricate pattern of switch requests across the timeline, involving multiple devices, often results in overlapping data points at the same timestamp—a challenge that standard time-series models find difficult to manage. Our event segmentation component, however, effectively handling overlaps from concurrent device requests. Second, standard models are constrained to using data from only the previous 24 hours due to input dimension limitations, potentially omitting less frequent devices that trigger requests on days. In contrast, our segmentation component utilizes multi-scale features extracted from data spanning the last day, week, month, and even year over an IoT device's entire lifecycle. This significantly enhancesthe detection of infrequent activities.

In addition, NeSIM demonstrates superior performance compared to the PatchTST + Segmentation approach. This improvement is primarily due to the advanced pre-



(a) Standard Models (b) Enhanced Models

Figure 3.12: Timing Forecast Accuracy. (a) Two standard time-series models; (b) Enhanced models with segmentation decoder.

training techniques employed by TimesFM, which endow the model with exceptional depth in pattern recognition capabilities.

3.4.2 Timing Forecast Accuracy

Since the timing forecast is the most critical task in NeSIM, we provide a detailed discussion of its accuracy. The accuracy of timing predictions is measured using the Mean Absolute Error (MAE) of the aligned timestamps between each predicted event series and the ground truth after applying the DTW algorithm. This metric evaluates the error in predicting the timings of event occurrences. The CDF of the timing MAE is illustrated in Fig. 3.12.

The standard time-series models achieve mean MAEs of 20.06 minutes (median: 22.08; P_{90}: 28.28) and 19.01 minutes (median: 17.52; P_{90}: 24.92), respectively. This implies that IoT devices would need to switch to a new operator network approximately 40 minutes in advance to account for potential forecast errors. However, this is impractical, as the device may not yet be within the coverage area of the new operator, especially considering that the current FOMOA process requires only 10 minutes.

In contrast, the enhanced models, PatchTST+Segmentation and NeSIM, achieve

significantly better mean MAEs of 3.42 (median: 2.07; P_{90}: 7.68) and 2.96 (median: 1.5; P_{90}: 7.17) minutes, respectively, outperforming their standard counterparts by 82.95% and 84.43%. NeSIM also shows a 13.5% improvement over the PatchTST+Segmentation model, consistent with earlier performances. With minute-scale precision for representing event occurrences, the theoretical minimum MAE is 1 minute. NeSIM approaches this near-perfect forecasting capability. To minimize connectivity disruptions caused by prediction inaccuracies, it is advisable to initiate the profile download to the device approximately six minutes before the predicted event time. These results fully demonstrate the feasibility and practical application of forecasting switch events using large AI models.

3.4.3 MOA Latency

Next, we estimated the latency caused by the existing FOMOA and the XiTuXi-driven LAMOA. The latency of FOMOA was computed based on the logs. For each daily forecast, latency was computed as follows: a correct prediction incurs a delay of c_s seconds and an incorrect prediction incurs c_f seconds, where c_s and c_f adhere to the distributions shown in Fig. 3.8 and Fig. 3.9, respectively. For a device $u \in \widehat{U} \cap U$ (i.e., true positives), the prediction is considered correct if the predicted event occurs within 6 minutes ahead of the ground truth and the operator prediction is accurate with c_s time; otherwise, the prediction fails and incurs c_f time. For a device $u \in \widehat{U} - U$ (i.e., false positives), all event predictions for this device are incorrect and incur the penalty c_f per event, as switches will be still arranged for these falsely predicted devices. For a device $u \in U - \widehat{U}$ (i.e., false negatives), the latency is computed as c_f for each missed event.

The resulting latency CDF is depicted in Fig. 3.13. FOMOA and LAMOA recorded mean latencies of 9.93 minutes (median: 8.7; P_{90}: 15.17) and 5.26 minutes (median: 4.18; P_{90}: 9.48), respectively. The latency observed for FOMOA in the test dataset

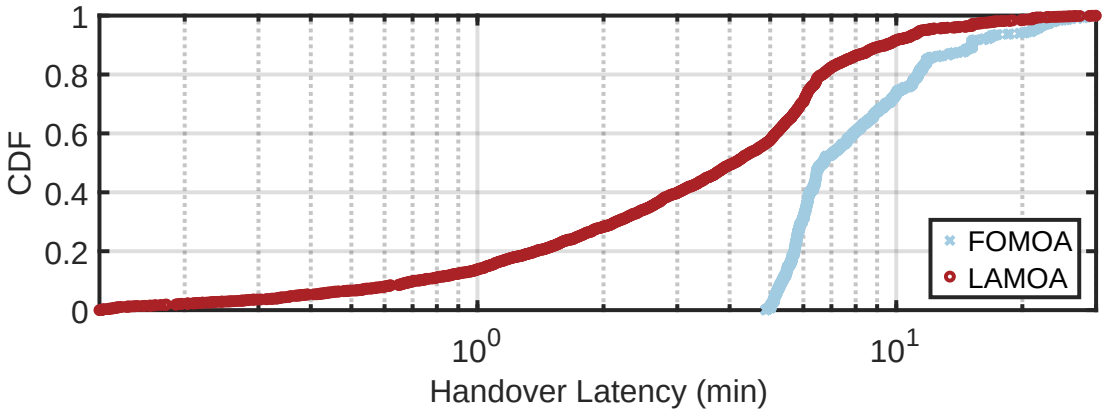


Figure 3.13: MOA Latency. This figure estimates the switching latency for the existing FOMOA and the NeSIM-driven LAMOA.

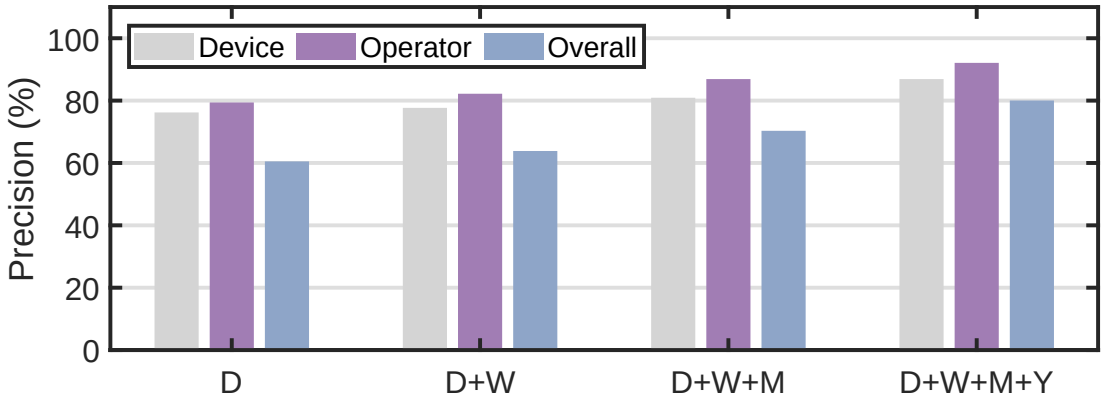


Figure 3.14: Impact of Historical Horizon. An ablation study is taken to consider the impact of historical horizon on the precision by removing the yearly, monthly and weekly historical data gradually.

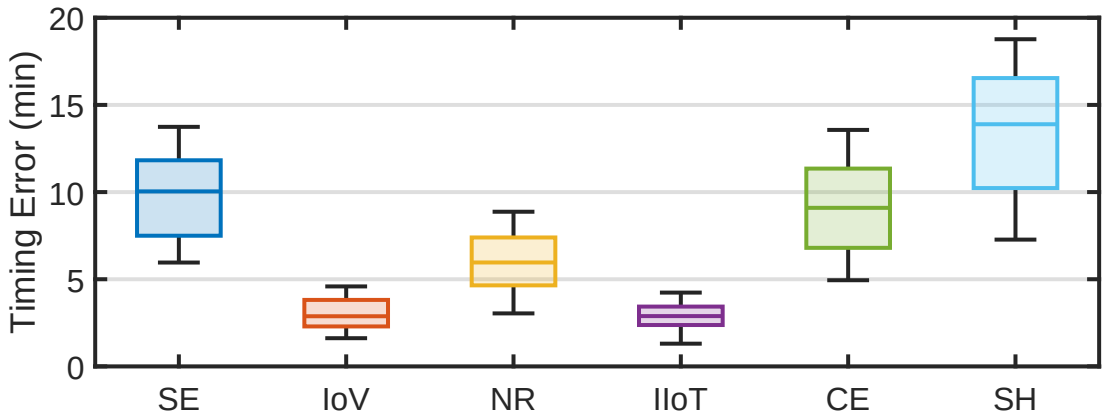


Figure 3.15: MAE of Timing Forecast across Industrial Sectors. These sectors include Sharing Economy (SE), Internet of Vehicles (IoV), New Retail (NR), Industrial IoT (IIoT), Consumer Electronics (CE), and Smart Healthcare (SH).

aligns closely with previously recorded measurements (see Fig. 3.9), while LAMOA reduces it by 51.95%. However, the latency still far exceeds the ideal 10-second switch time, primarily due to penalties from 19.7% false positives and 30.36% false negatives. Despite these challenges, LAMOA significantly reduces latency by half compared to FOMOA, highlighting notable performance improvements.

3.4.4 Impact of Historical Horizon

To validate the efficacy of the multi-scale features extracted by the backbone encoder, we conducted an ablation study using four different configurations. For each configuration, we tested the model by filling missing scale features with zeros. We then assessed the impact of these configurations on two classification tasks. The results are presented in Fig. 3.14. Regardless of whether the forecast was for device or operator, the trends in precision were consistent: precision improved with the inclusion of longer historical records. For example, the overall precision increased from 50.79% to 79.64%. These results substantiate the effectiveness of employing multi-scale feature extraction in improving forecast accuracy.

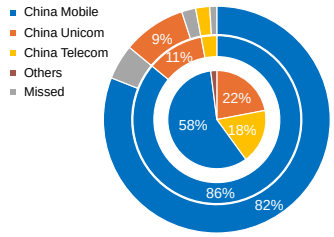


Figure 3.16: Accuracy across Operators

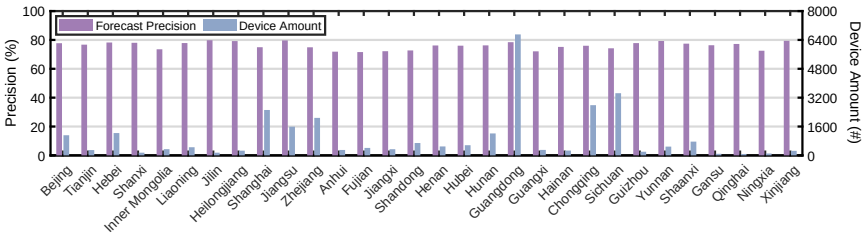


Figure 3.17: Forecast Accuracy across Nation

3.4.5 Performance across Sectors

Next, we evaluate the timing forecast performance of NeSIM across various industrial sectors. The distribution of MAE by sector is illustrated in Fig. 3.15, which displays the MAE values across six key sectors, whose switch frequencies range from under 5 to 30 times per day. As indicated by the figure, the sectors with the lowest MAEs —and thus better timing accuracy—are the Internet of Vehicles and Industrial IoT. These sectors demonstrate more precise forecast performance, likely due to more consistent and predictable usage patterns within these industries. New Retail also shows relatively good forecast accuracy, positioned closely behind the leading sectors. In contrast, Smart Healthcare exhibits the highest MAE, suggesting challenges in timing accuracy possibly due to the complexity and unpredictability of healthcare operations Another factor contributing to the higher performance in IIoT, IoV, and NR is their larger share of the dataset, accounting for 67% (see Fig. 3.5), which provides more historical records for pattern learning.

3.4.6 Performance across Operators

We further examine the forecast missing rate (i.e., 1-Recall) across operators. The distributions are presented in Fig. 3.16. The innermost pie chart shows the market shares of the operators, with China Mobile, China Unicom, and China Telecom holding 56%, 23%, and 18% of the market [53], respectively. The middle pie chart illustrates the distribution of true switch requests processed through our platform,

with 86%, 11%, and 3% of requests switching to the three operators, respectively. This distribution closely aligns with the market shares. The outermost pie chart quantifies the forecast accuracy, revealing that NeSIM missed 4%, 2%, and 1% of the total switch events for each operator, respectively. This indicates that the system performs slightly better for operators with smaller market shares, likely due to fewer switch events and less variability in their coverage patterns.

3.4.7 Performance across the Nation

Finally, we assess the forecast performance of our eSIM service across the 30 administrative regions in China. The precision of device forecasts and their amount regarding the regions are depicted in Fig. 3.17. While the results are generally consistent across the regions, the 4 municipalities and economically significant provinces show slightly higher precision. This improved accuracy can be attributed to their large populations, which lead to a denser network of base stations and fuller coverage. Additionally, the high demand for IoT devices in these areas provides a larger volume of historical data, which enhances the forecasting accuracy.

3.5 Related Work

Our work is related to the following three fields:

eSIM Technology. Recent advancements in eSIM technology have enhanced flexibility, security, and efficiency in cellular networks [54, 55, 56, 57, 58, 59, 60, 61, 62]. Studies highlight the growing demand for eSIM due to its remote provisioning capabilities and seamless carrier switching [54], while its role in 5G new radio evolution is discussed in [55]. eSIM integration enhances QoS for critical applications [56], and security challenges related to deployment are addressed with improved authentication mechanisms [57].

Multi-Operator Access. Recent studies utilizing eSIM technology for multi-carrier and multi-operator access have aimed to improve network reliability, security, and flexibility [63, 64, 65, 66]. Current RSP solutions, both operator and OEM-controlled, often use a universal profile that can connect to any network but costs 5 to 10 times more than a single-operator profile [67]. While suitable for high-end cellular phones, this cost makes universal profiles impractical for cost-sensitive IoT devices. Alternative strategies, like those proposed by MOTA [67] and iCelluar [68], recommend that devices scan and choose networks based on QoS, RSS, and mobility patterns for optimal network selection. However, the feasibility of this method in IoT is limited due to significant challenges: network scanning disrupts transmissions; and model running increases energy consumption.

Time-Series Forecasting. Recent advancements in time-series forecasting have focused on enhancing model accuracy, scalability, and adaptability across various domains [69, 70, 71, 49, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91]. Several studies explore novel approaches such as the TTM, which provide efficient pre-trained models for multivariate time-series forecasting [69]. Transformer-based models like PatchTST have been extensively evaluated for their decoder-only foundation architecture [49]. Implicit reasoning and contextual modeling approaches have also been explored to enhance forecasting accuracy in various applications [72, 75]. These advancements collectively contribute to the ongoing evolution of time-series forecasting, enhancing its applicability and reliability in real-world scenarios.

3.6 Conclusion

This study pioneers the large-scale evaluation of eSIM-enabled mobile IoT, demonstrating its significant role in enhancing the scalability of mobile IoT systems. In addition, we introduce NeSIM, the first design and implementation of the LAMOA strategy, which minimizes disconnectivity risks and reduces handover latency.

Chapter 4

Conclusions and Future Work

4.1 Conclusions

In conclusion, this dissertation has explored the transformative potential of large AI models in wireless systems, focusing on two critical areas: cross-technology communication (CTC) and intelligent multi-operator access (MOA) in IoT networks. By leveraging the power of neural machine translation (NMT) and advanced time-series forecasting, we have developed innovative solutions that address the limitations of traditional approaches and pave the way for intelligent and adaptive connectivity. Specifically, we make the following contributions:

- *Neural Machine Translation for Cross-Technology Communication:* This work, XiTuXi, represents a paradigm shift in cross-technology communication by introducing a neural machine translation framework that automates the process of translating between heterogeneous wireless protocols. By treating wireless protocols as translatable languages, XiTuXi eliminates the need for reverse engineering and enables seamless communication between WiFi and various IoT protocols such as ZigBee, Bluetooth, and LoRa. This approach not only reduces

the symbol error rate (SER) and improves the frame reception rate (FRR) but also scales efficiently across 30 protocol combinations, significantly outperforming previous solutions. The success of XiTuXi demonstrates the potential of AI-driven frameworks to break down protocol silos and enable interoperability in diverse wireless environments.

- *Lookahead Multi-Operator Access for IoT Networks:* This work introduces NeSIM, a novel framework designed to address the challenges of multi-operator access in large-scale IoT networks by leveraging predictive analytics to optimize operator switch events. Through the use of a pre-trained time-series model, NeSIM forecasts connectivity disruptions and proactively schedules operator switches, significantly minimizing handover latency and reducing the risk of disconnectivity. Evaluated on a comprehensive dataset comprising 30,000 IoT devices and 16.95 million historical switch records, NeSIM achieves a 47% reduction in latency and an 80.63% decrease in disconnectivity risk, showcasing its scalability and effectiveness in real-world deployments. This work underscores the transformative potential of AI-driven predictive strategies in enhancing the reliability and efficiency of eSIM-enabled IoT networks.

4.2 Lessons Learned

This research journey has been both challenging and rewarding, offering valuable insights into the process of conducting robust and impactful research. Through the development of XiTuXi and NeSIM, I learned not only about the technical aspects of integrating AI into wireless systems but also about the broader principles of effective research. These lessons span from problem definition and iterative development to the importance of interdisciplinary collaboration, each of which has shaped my approach to tackling complex problems and driving innovation.

Clear Problem Definition. One of the most critical lessons is the necessity of clearly defining the problem and its scope before diving into solutions. In both XiTuXi and NeSIM, the initial challenge was to identify the core limitations of existing approaches and articulate how AI-driven methods could address them. For example, in XiTuXi, the problem of cross-technology communication was framed as a language translation task, which allowed us to leverage neural machine translation techniques effectively. Similarly, in NeSIM, the focus on predictive analytics for multi-operator access emerged from a thorough analysis of the inefficiencies in failover-based strategies. This process taught me that a well-defined problem not only guides the research direction but also ensures that the solutions are both impactful and feasible. Moreover, it highlights the importance of balancing ambition with practicality—while it' s tempting to tackle broad challenges, narrowing the scope often leads to more robust and actionable results.

Iterative Development. Another key lesson is the value of iterative development and rigorous validation. In both projects, the initial prototypes were far from perfect, and it was through continuous testing, refinement, and feedback that the final frameworks achieved their goals. For instance, in XiTuXi, the initial translation model struggled with symbol misalignment, which was resolved by introducing a genetic algorithm to generate a robust training corpus. Similarly, in NeSIM, the time-series forecasting model underwent multiple iterations to improve its accuracy and scalability. This iterative process taught me that robust research is not about getting it right the first time but about learning from failures and adapting accordingly. It also reinforced the importance of validation—whether through simulations, real-world datasets, or user feedback—to ensure that the solutions are not only theoretically sound but also practically viable.

Interdisciplinary Collaboration. A third lesson is the immense value of interdisciplinary collaboration and diverse perspectives in driving innovation. Both XiTuXi and NeSIM required expertise from multiple domains, including wireless communica-

tion, machine learning, and data analytics. For example, XiTuXi combined insights from signal processing and natural language processing to translate wireless protocols, while NeSIM integrated time-series forecasting techniques with network optimization strategies. Collaborating with experts from different fields not only enriched the research but also helped identify blind spots and uncover novel solutions. This experience taught me that groundbreaking research often emerges at the intersection of disciplines, and fostering a collaborative environment is essential for tackling complex problems. It also highlighted the importance of effective communication—translating technical concepts across domains to ensure alignment and shared understanding.

4.3 Future Work

The advancements achieved by XiTuXi and NeSIM have laid a strong foundation for addressing key challenges in wireless systems through the transformative potential of AI. However, as wireless networks continue to evolve, new opportunities and challenges emerge that call for further exploration. These future directions aim to build on the successes of XiTuXi and NeSIM, leveraging cutting-edge technologies such as federated learning, edge AI, and spectrum-sharing ecosystems to advance the intelligence, adaptability, and scalability of wireless networks.

4.3.1 Federated Learning for Distributed Optimization in Wireless Networks

While NeSIM demonstrated the effectiveness of predictive analytics in optimizing multi-operator access, its reliance on centralized data processing poses scalability and privacy challenges. Future work could explore the application of federated learning, where IoT devices collaboratively train a shared model without sharing raw data. This approach would not only preserve user privacy but also enable real-time adaptation to

local network conditions. Key challenges include designing efficient communication protocols to minimize overhead and ensuring model convergence in heterogeneous environments. By integrating federated learning, future frameworks could achieve distributed optimization while maintaining scalability and robustness.

4.3.2 Edge AI for Low-Latency Decision-Making in Cross-Technology Communication

XiTuXi successfully automated cross-technology communication using neural machine translation, but its reliance on cloud-based processing introduces latency that may not be acceptable for time-sensitive applications. Future work could focus on deploying lightweight AI models at the network edge to enable low-latency decision-making. This would involve optimizing model architectures for resource-constrained devices, such as using quantization or pruning techniques, and developing edge-compatible training pipelines. Additionally, exploring real-time adaptation mechanisms to handle dynamic channel conditions could further enhance the reliability of cross-technology communication in edge environments.

4.3.3 Spectrum-Sharing Ecosystems for Next-Generation Wireless Networks

As wireless networks evolve toward 6G and beyond, the demand for efficient spectrum utilization will intensify. Building on the insights from XiTuXi and NeSIM, future research could explore the development of AI-driven spectrum-sharing ecosystems. These ecosystems would enable dynamic allocation of spectrum resources among heterogeneous networks (e.g., 5G, WiFi, IoT) based on real-time demand and interference conditions. Key challenges include designing fair and incentive-compatible sharing mechanisms, ensuring compliance with regulatory constraints, and integrat-

ing AI models for predictive spectrum management. Such ecosystems could unlock new levels of efficiency and scalability, paving the way for ubiquitous connectivity in the next generation of wireless networks.

References

[1] Zhijun Li and Tian He. Webee: Physical-layer cross-technology communication via emulation. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 2–14, 2017.

[2] Zhijun Li and Tian He. Longbee: Enabling long-range cross-technology communication. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 162–170. IEEE, 2018.

[3] Yongrui Chen, Zhijun Li, and Tian He. Twinbee: Reliable physical-layer cross-technology communication with symbol-level coding. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 153–161. IEEE, 2018.

[4] Ruofeng Liu, Zhimeng Yin, Wenchao Jiang, and Tian He. Wibeacon: Expanding ble location-based services via wifi. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 83–96, 2021.

[5] Hsun-Wei Cho and Kang G Shin. Bluefi: bluetooth over wifi. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 475–487, 2021.

[6] Dan Xia, Xiaolong Zheng, Fu Yu, Liang Liu, and Huadong Ma. Wira: Enabling cross-technology communication from wifi to lora with ieee 802.11 ax. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 430–439. IEEE, 2022.

[7] Xiuzhen Guo, Yuan He, Jia Zhang, and Haotian Jiang. Wide: Physical-level ctc via digital emulation. In *2019 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 49–60. IEEE, 2019.

[8] Yuanhe Shu, Jingwei Wang, Linghe Kong, Jiadi Yu, Guisong Yang, Yueping Cai, Zhen Wang, and Muhammad Khurram Khan. Wibwi: Encoding-based bidirectional physical-layer cross-technology communication between ble and wifi. In *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 356–363. IEEE, 2021.

[9] Xin Na, Xiuzhen Guo, Yuan He, and Rui Xi. Wi-attack: Cross-technology impersonation attack against ibeacon services. In *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2021.

[10] Hsun-Wei Cho and Kang G Shin. Flew: fully emulated wifi. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 29–41, 2022.

[11] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[15] Chatgpt. https://openai.com/blog/chatgpt/.

[16] Lora matlab. https://github.com/bhomssi/LoRaMatlab.

[17] Zhijun Li and Yongrui Chen. Bluefi: Physical-layer cross-technology communication from bluetooth to wifi. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 399–409. IEEE, 2020.

[18] Openwrt. https://openwrt.org/.

[19] Gl-ar750s: Gigabit wireless router for security-savvy travelers. https://www.gl-inet.com/products/gl-ar750s/.

[20] Launchxl-cc1352p1: Simplelink multi-band cc1352p wireless mcu launchpad development kit. https://www.ti.com/tool/LAUNCHXL-CC26X2R1/.

[21] Launchxl-cc26x2r1: Simplelink multi-standard cc26x2r wireless mcu launchpad development kit. https://www.ti.com/tool/LAUNCHXL-CC26X2R1/.

[22] Pycom sipy sigfox development board. https://docs.pycom.io/datasheets/development/sipy/.

[23] Insoo Chung, Byeongwook Kim, Yoonjung Choi, Se Jung Kwon, Yongkweon Jeon, Baeseong Park, Sangha Kim, and Dongsoo Lee. Extremely low bit transformer quantization for on-device neural machine translation. *arXiv preprint arXiv:2009.07453*, 2020.

[24] Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*, 2020.

[25] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on machine learning*, pages 5958–5968. PMLR, 2020.

[26] Xiuzhen Guo, Yuan He, Xiaolong Zheng, Liangcheng Yu, and Omprakash Gnawali. Zigfi: Harnessing channel state information for cross-technology communication. *IEEE/ACM Transactions on Networking*, 28(1):301–311, 2020.

[27] Kameswari Chebrolu and Ashutosh Dhekne. Esense: Communication through energy sensing. In *Proc. of ACM MobiCom*, 2009.

[28] Song Min Kim and Tian He. Freebee: Cross-technology communication via free side-channel. In *Proc. of ACM MobiCom*, 2015.

[29] Zicheng Chi, Yan Li, Hongyu Sun, Yao Yao, Zheng Lu, and Ting Zhu. B2w2: N-way concurrent communication for iot devices. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems*, pages 245–258, 2016.

[30] Yongrui Chen, Shuai Wang, Zhijun Li, and Tian He. Reliable physical-layer cross-technology communication with emulation error correction. *IEEE/ACM Transactions on Networking*, 28(2):612–624, 2020.

[31] Yifan Zhang and Qun Li. Howies: A holistic approach to zigbee assisted wifi energy savings in mobile devices. In *Proc. of IEEE INFOCOM*, 2013.

[32] Zhimeng Yin, Wenchao Jiang, Song Min Kim, and Tian He. C-morse: Cross-technology communication with transparent morse coding. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[33] Wenchao Jiang, Zhimeng Yin, Song Mim Kim, and Tian He. Transparent cross-technology communication over data traffic. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[34] Wenchao Jiang, Zhimeng Yin, Ruofeng Liu, Zhijun Li, Song Min Kim, and Tian He. Bluebee: a 10,000 x faster cross-technology communication via phy emulation. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, pages 1–13, 2017.

[35] Hsun-Wei Cho and Kang G Shin. Drew: Double-throughput emulated wifi. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pages 663–678, 2024.

[36] Hsun-Wei Cho. *Enabling Direct Bluetooth-WiFi Communications*. PhD thesis, 2024.

[37] Junyang Shi, Di Mu, and Mo Sha. Lorabee: Cross-technology communication from lora to zigbee via payload encoding. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, pages 1–11. IEEE, 2019.

[38] Shuoheng Yang, Yuxin Wang, and Xiaowen Chu. A survey of deep learning techniques for neural machine translation. *arXiv preprint arXiv:2002.07526*, 2020.

[39] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*, 2018.

[40] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR, 2017.

[41] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[42] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

[43] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[44] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

[45] GSMA. Mobile IoT: Enabling the internet of things, 2023.

[46] GSMA. esim iot technical specification. Technical specification, GSMA, April 2024.

[47] BankMyCell. Average lifespan of smartphones. https://www.bankmycell.com/blog/average-lifespan-of-smartphone, 2023.

[48] The State Council of the People's Republic of China. Statistical report on china's telecommunications infrastructure. https://english.www.gov.cn/archive/statistics/202401/19/content_WS65aa171cc6d0868f4e8e34c5.html, 2024.

[49] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023.

[50] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

[51] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[52] Google Research. Timefm: A pretrained foundation model for time series forecasting. https://github.com/google-research/timesfm, 2023.

[53] TeleGeography. China: The world's biggest 5g market, 2023.

[54] I. Godlovitch, S. Martins, C. Gries, J. Knips, and C. Wernick. Study on wholesale mobile connectivity, trends and issues for emerging mobile technologies and deployments. 2023.

[55] B. Abazi. *5G new radio evolution towards enhanced features*. PhD thesis, University of Oulu, 2023.

[56] E. Obiodu. *Differentiated QoS connectivity for society-critical use cases in the 5G era*. PhD thesis, King' s College London, 2023.

[57] C. Casetti. 5g consolidates deployment by targeting new bands. *IEEE Vehicular Technology Magazine*, 2021.

[58] A. Al Mousa, M. Al Qomri, and S. Al Hajri. Utilizing the esim for public key cryptography: A network security solution for 6g. In *IEEE Conference Proceedings*, 2020.

[59] J. Zhao, B. Ding, Y. Guo, Z. Tan, and S. Lu. Securesim: Rethinking authentication and access control for sim/esim. In *ACM International Conference on Mobile Computing and Networking*, 2021.

[60] T. Yoshizawa and B. Preneel. Verification schemes of multi-sim devices in mobile communication systems. In *ACM Symposium on Mobility Management and Wireless Access*, 2020.

[61] M. Echeverria. *Applying Automated Reasoning to Analyze and Protect Cellular Network Protocols*. PhD thesis, ProQuest, 2023.

[62] S. A. Syverud. Security of 5g-enabled next generation emergency communication in norway. *NTNU Open*, 2020.

[63] A. Gomes, J. Kibiłda, and A. Farhang. Multi-operator connectivity sharing for reliable networks: A data-driven risk analysis. In *IEEE International Conference on Network and Service Management*, 2021.

[64] E. Obiodu, A. Raman, and A.K. Abubakar. Dsm-moc as baseline: Reliability assurance via redundant cellular connectivity in connected cars. In *IEEE International Conference on Network and Service Management*, 2022.

[65] M. Sauter. *Long Term Evolution (LTE) und LTE-Advanced*. Springer, 2022.

[66] J.T.J. Penttinen. *Wireless Communications Security: Solutions for the Internet of Things*. Springer, 2016.

[67] Supratim Deb, Kanthi Nagaraj, and Vikram Srinivasan. Mota: Engineering an operator agnostic mobile service. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, pages 133–144, 2011.

[68] Yuanjie Li, Haotian Deng, Chunyi Peng, Zengwen Yuan, Guan-Hua Tu, Jiayao Li, and Songwu Lu. icellular:device-customized cellular network access on commodity smartphones. In *In Proc. of NSDI 16*, pages 643–656, 2016.

[69] V. Ekambaram, A. Jati, P. Dayama, and S. Mukherjee. Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. *CoRR*, 2024.

[70] Y. Hou, C. Ma, X. Li, Y. Sun, H. Yu, and Z. Fang. Time series foundation model for improved transformer load forecasting and overload detection. *Preprints.org*, 2024.

[71] H.K. Saravanan, S. Dwivedi, and P. Praveen. Analyzing the performance of time series foundation models for short-term load forecasting. In *ACM Proceedings*, 2024.

[72] W. Potosnak, C. Challu, M. Goswami, and M. Wiliński. Implicit reasoning in deep time series forecasting. *arXiv*, 2024.

[73] A.V. Andrei, G. Velev, F.M. Toma, and D.T. Pele. Energy price modelling: A comparative evaluation of four generations of forecasting methods. *arXiv*, 2024.

[74] N. Huang, H. Wang, Z. He, M. Zitnik, and X. Zhang. Repurposing foundation model for generalizable medical time series classification. *arXiv*, 2024.

[75] S. Chattopadhyay, P. Paliwal, and S.S. Narasimhan. Context matters: Leveraging contextual features for time series forecasting. *arXiv*, 2024.

[76] L.H. Anh, D.T. Vu, S. Oh, G.H. Yu, and N.B.N. Han. Partial transfer learning from patch transformer to variate-based linear forecasting model. *Energies*, 2024.

[77] L. Yang, Y. Wang, X. Fan, I. Cohen, and J. Chen. Vitime: A visual intelligence-based foundation model for time series forecasting. *arXiv*, 2024.

[78] J. Zhang, S. Zheng, X. Wen, X. Zhou, and J. Bian. Elastst: Towards robust varied-horizon forecasting with elastic time-series transformer. *arXiv*, 2024.

[79] Z. Li, X. Qiu, P. Chen, Y. Wang, H. Cheng, and Y. Shu. Foundts: Comprehensive and unified benchmarking of foundation models for time series forecasting. *arXiv*, 2024.

[80] A. Das, M. Faw, R. Sen, and Y. Zhou. In-context fine-tuning for time-series foundation models. *arXiv*, 2024.

[81] T. Aksu, G. Woo, J. Liu, X. Liu, C. Liu, and S. Savarese. Gift-eval: A benchmark for general time series forecasting model evaluation. *arXiv*, 2024.

[82] M. Meyer, D. Zapata, S. Kaltenpoth, and O. Müller. Benchmarking time series foundation models for short-term household electricity load forecasting. *arXiv*, 2024.

[83] M. Chen, L. Shen, Z. Li, X.J. Wang, J. Sun, and C. Liu. Visionts: Visual masked autoencoders are free-lunch zero-shot time series forecasters. *arXiv*, 2024.

[84] C. Feng, L. Huang, and D. Krompass. General time transformer: An encoder-only foundation model for zero-shot multivariate time series forecasting. *ACM Proceedings*, 2024.

[85] H. Hu, Z. Han, S. Qian, D. Yang, and J. Cao. Pattern-oriented attention mechanism for multivariate time series forecasting. *ACM Transactions on Knowledge Discovery from Data*, 2025.

[86] Y. Yu. *Physics-Informed Data-driven Modeling of Complex Networked Systems through Graph Neural Networks*. PhD thesis, Pennsylvania State University, 2025.

[87] Y. Ma, A. Patel, D. Kurian, J. Siebert, S. Vock, and A. Morozov. A time-series data generation tool for risk assessment of robotic applications. *ResearchGate*, 2024.

[88] G. Wang, Z. Wu, and B. Liang. Slicing-free supervised dimension reduction for multivariate time series. *Statistics*, 2025.

[89] D. Ge, B. Liu, Y. Cheng, and Z. Zhu. Rethinking decomposition for time series forecasting: Learning from temporal pattern. *SSRN*, 2024.

[90] J. Carmo de Almeida Neto and C. Miceli de Farias. Forecasting telemetry data using gan and bilstm in a digital twin. *arXiv*, 2025.

[91] A. Elshamy and H. Salem. Long-term forecasting of climate variability using hybrid deep learning models. *arXiv*, 2024.