Toward Semantic Communication Protocols: A Probabilistic Logic Perspective

Sejin Seo, Graduate Student Member, IEEE, Jihong Park, Senior Member, IEEE, Seung-Woo Ko, Senior Member, IEEE, Jinho Choi, Fellow, IEEE, Mehdi Bennis, Fellow, IEEE, and Seong-Lyun Kim

Abstract—Classical medium access control (MAC) protocols are interpretable, yet their task-agnostic control signaling messages (CMs) are ill-suited for emerging mission-critical applications. By contrast, neural network (NN) based protocol models (NPMs) learn to generate task-specific CMs, but their rationale and impact lack interpretability. To fill this void, in this article, we propose, for the first time, a semantic protocol model (SPM) constructed by transforming an NPM into an interpretable symbolic graph written in the probabilistic logic programming language (ProbLog). This transformation is viable by extracting and merging common CMs and their connections, while treating the NPM as a CM generator. By extensive simulations, we corroborate that the SPM tightly approximates its original NPM while occupying only 0.02% memory. By leveraging its interpretability and memory-efficiency, we demonstrate several SPM-enabled applications such as SPM reconfiguration for collision-avoidance, as well as comparing different SPMs via semantic entropy calculation and storing multiple SPMs to cope with non-stationary environments.

Index Terms—Semantic communication protocol, protocol learning, medium access control (MAC), probabilistic logic programming language (ProbLog), semantic information theory, multi-agent deep reinforcement learning.

I. INTRODUCTION

The advent of diverse service verticals in 6G introduces a variety of unique and stringent service requirements [1]. This poses a significant challenge in developing a sustainable communication system using current one-size-fits-all and standardization-based approaches. As an alternative, semantic communication is an emerging paradigm that designs a communication system inherently oriented towards a given task and empowered by task-specific semantics through machine learning [2]. While recent progress has been made in semantic communication, most studies have focused on physical layer (PHY) designs [3]. In this article, we explore the potential of this semantic communication paradigm in designing a communication protocol in the medium access control (MAC) layer, which facilitates communication and coordination between devices within a network. To better understand how MAC protocols can address the evolving needs of 6G in this new design paradigm, we categorize them into three levels.

A. Level 1 – Human-Crafted Classical Protocol

Level 1 protocols represent classical MAC protocols that are hand-crafted through standardization activities [4], [5]. Designed for general purposes, these protocols lack adaptability to changing environments. While scheduling policies and handshaking rules can be modified to some extent (e.g., grant-free access prioritization [6]), their control signaling messages (CMs) remain static, even when tasks and other environmental characteristics change over time. This lack of flexibility presents a significant obstacle for emerging 6G applications, which are mission-critical under non-stationary environments, such as drones and satellites in non-terrestrial networks (NTNs) [7], autonomous cars in vehicle-to-everything (V2X) networks [8], and visuo-haptic immersive applications in the metaverse [9]. These applications demand MAC protocols that can be customized to their specific task characteristics and requirements.

B. Level 2 – Task-Oriented Neural Protocol

In contrast to Level 1 protocols that are pre-programmed and general-purpose, Level 2 protocols are learned directly from a given task-specific environment [10], [11], [12]. For the sake of explanation without loss of generality, throughout this paper we focus on the protocol learning scenario studied by Mota et al. [10], which consists of user equipments (UEs) associated with a base station (BS). The task is to establish collision-free data communication in the user plane by UEs’ taking communication actions after exchanging CMs with the BS in the control plane, as depicted in Fig. 1. Following the approach proposed in [13], this task is formulated as a multi-agent deep reinforcement learning (MADRL) problem.
as done in [10]. In this formulation, as illustrated in Fig. 2, each single-cycle MADRL operation can be recast as the forward propagation (FP) of a neural network (NN) from input to output, where the input is the UEs’ states (i.e., buffer states), and the output is their actions (i.e., access or silence). As the NN training converges via MADRL, the uplink CMs (UCMs) and downlink CMs (DCMs) become meaningful CMs that enable task-optimal actions. The trained NN can then be considered as a neural protocol model (NPM) that emerged from the task-specific environment. However, NPMs have several drawbacks due to their NN architecture. In fact, the NPM is a black-box function with limited interpretability, and its parameters cannot be immediately reconfigured, thus reducing flexibility in protocol operations. Furthermore, the NPM is over-parameterized as an NN, which entails large communication payloads, computing time, and memory usage.

C. Level 3 – Task-Oriented Semantic Protocol

In this article, we introduce Level 3 protocols designed to overcome the limitations of Level 2 NPMs while maintaining their task-oriented benefits. Our primary strategy involves developing a semantic protocol model (SPM) by extracting semantics from an NPM. We adopt the definition of semantics from the semantic web [14], which define semantics as *concepts* and their *connections* that remain invariant. In an NPM, these concepts are represented by the UCMs and DCMs, and their connections are determined by the NPM’s FP structure, i.e., state→UCM→DCM→action. To construct an SPM, we propose a novel NPM-to-SPM transformation technique.

Initially, we feed each state experienced during training to an NPM, which generates a rule determining each single-cycle protocol operation of state→UCM→DCM→action. By collecting all generated rules, we create an NPM extract, as shown in Fig. 3. Since NPM training is task-oriented and does not account for semantics, the NPM extract contains redundant concepts and connections that can be merged to reduce the number of CMs and the size of the SPM. However, this merging can lead to inconsistent rules. If we treat UCMs/DCMs as *vocabulary*, this inconsistency resembles the problem of *polysemy* (a word with multiple meanings) in linguistics [15], [16]. Drawing inspiration from how humans differentiate the meanings of polysemous words based on context, we assign a context \( p \in [0, 1] \) to each connection “→” of a rule, which quantifies the generation frequency of the concept within the NPM.

To facilitate reconfiguration, we establish a basic manipulable unit of an SPM as each connection of two neighboring vocabularies \( A \) and \( B \) with its frequency context \( p \). We represent this \( A \xrightarrow{p} B \) as a *clause* \( p :: B \,: A \) written in the probabilistic logic programming language (ProbLog) [17], a logic programming language widely used in the field of symbolic artificial intelligence (AI) [18]. A set of these probabilistic clauses forms a rule determining each single-cycle protocol operation. The aggregation of these rules constitutes an SPM.

Consequently, an SPM is interpretable and reconfigurable, allowing, for example, human operators to implement immediate control of CA without re-training. Moreover, the semantic uncertainty of each SPM can be measurable by evaluating the average entropy of all clauses, enabling to compare multiple SPMs and select the most suitable SPM for a specific environment. As Table I demonstrates, the ProbLog format of this SPM occupies only 0.02% of the memory compared to its original NPM. This not only accelerates protocol operations by 1,750x in terms of floating point operations per second (FLOPS), but also permits the storage of a portfolio of multiple SPMs for adapting to non-stationary environments.

D. Backgrounds and Related Works

The semantic communication paradigm shares significant connections with AI-native communication system designs [11], [19], [20]. We review the PHY and MAC layer perspectives of these approaches, and highlight their similarities and differences concerning semantic definitions and extraction methods, as elaborated next.

1) Semantic PHY Designs: Recent studies on semantic communication have primarily focused on PHY designs [2], [3], with their foundations tracing back to the three levels of communication identified by W. Weaver [21]. Classical source and channel coding in Shannon’s communication framework [22] fall under Level A, which focuses on the technical communication of bits. On the other hand, Level B aims to deliver the semantics of the bits, and Level C is in pursuit of maximizing the effectiveness of delivering the semantics in a given task [21]. This semantics-and-effectiveness problem has long been overlooked due to the lack of tools, but recent advances in ML have helped bring it to the forefront.

In the context of computer vision [23], semantics refer to concepts that remain consistent across different source data samples. In source-oriented rate-distortion theory [24], semantics refer to the generative process of source data samples. Recent semantic PHY designs for Level B, such as deep learning based semantic communication (DeepSC) [25] and deep learning based joint source and channel coding (DeepJSCC) frameworks [26], follow these definitions, and use autoencoder (AE) to extract semantics from source data. An AE is an NN that encodes an input \( X \) into a latent representation \( Z \), which is then decoded to reconstruct \( X \) (i.e., \( X \rightarrow Z \rightarrow X \)). While a standard AE aims to compress \( X \) into \( Z \) with a smaller dimension, semantic PHY designs leverage an AE to find \( Z \) that is invariant to changes in source data samples from \( X \) to \( X' \), even with different data types (i.e., \( \{X, X'\} \rightarrow Z \rightarrow \{X, X'\} \)). The decoding of \( Z \rightarrow \{X, X'\} \) implies that \( Z \) can generate the source data \( X \) and \( X' \), satisfying the aforementioned semantics definition.

Existing studies in this direction consider various data types such as text [27], speech [28], and image data [29].
Fig. 2. Schematic illustrations and KPI performance of the protocol models: (a) NPM; (b) graphical representation of SPM, and the number on the edges denotes the clause truth probability; and (c) ProbLog representation of SPM.

Fig. 3. Vocabularies and connections extracted from an NPM when $N = 2$ and $|B_1| = |B_2| = 3$. The bold arrows and boxes indicate the vocabularies being activated with the current input $b^{(1)}_1$ and $b^{(2)}_2$. The inference result makes UE 1 discards whereas UE 2 accesses the channel.

**TABLE I**

| Protocol | Goodput (Packets/Cycle) | UCM Vocabularies | DCM Vocabularies | UCM/DCM Length (Bytes) | Model Storage (Bytes) | Inference (FLOPs) | Collision (Collisions/Cycle) |
|----------|-------------------------|------------------|------------------|-------------------------|-----------------------|-------------------|-----------------------------|
| NPM      | 0.729 (100%)            | 10 (40%)         | 50 (6%)          | 32                      | 4.55M                 | 14K               | 0.275                       |
| SPM      | 0.729                   |                  |                  |                         |                       | 8                 | 0.0                         |

They also consider multi-modal [30] and cross-modal data applications [31]. Furthermore, this approach can also be extended to address a generic task for Level C by representing the output as a function of $X$ (i.e., $X \rightarrow Z \rightarrow f(X)$). For instance, $f(X)$ can be the human-perceived quality of raw data $X$ [32] or other non-reconstruction tasks such as control decision-makings [33].

2) **Semantic MAC Designs**: In wireless networks, MAC protocols coordinate multi-UE access and prevent collisions by exchanging CMs. Traditional MAC protocols, known as Level 1, rely on human-engineered and pre-defined CMs. Despite their success, these conventional methods face challenges due to the increasing complexity of MAC tasks and diverse access prioritization issues, such as those encountered in dynamic spectrum access [34], [35] and mission-critical applications [36], [37]. Alternatively, Level 2 introduces an NPM that exchange CMs emerged directly from multi-UE interactions within a task-specific environment via MADRL [10], [11], [12]. This approach is especially promising for addressing non-stationary environments like...
autonomous vehicles [38] and non-terrestrial networks [7], [39]) by optimizing CMs according to environmental changes. NPM operations for a UE are described as a sequence \( X_t \rightarrow CM_t \rightarrow f(X_t, CM_t) \rightarrow X_{t+1} \rightarrow \cdots \), with the UE’s state \( X_t \) at time \( t \) and the its access decision-making \( f(X_t, CM_t) \).

Building the NPM, Level 3 presents an SPM that is developed by extracting and representing the semantics of CMs in the form of a symbolic graph or equivalently a set of reconfigurable logic clauses, as depicted in Figs. 2(b) and (c), respectively. Similar to the importance of the latent representation \( Z \) in semantic PHY designs, the SPM focuses on CM semantics to enable interpretable and reconfigurable protocol operations while maintaining the task-oriented benefits of the NPM. The semantics definition of these CMs is more extensive than those in semantic PHY designs. For instance, \( CM_t \) remains invariant to insignificant state changes and can generate \( X_{t+1} \), satisfying the invariant and generative concept-based semantics definition found in semantic PHY designs at Levels B and C [24], [25], [26]. Moreover, inspired by the semantic web [14] and semantic information theory (SIT) [40], [41], semantic MAC designs additionally consider inter-CM connections and their uncertainties as CM semantics, reflecting multi-UE interactions and representing their causes and effects for interpretable protocol reconfiguration.

The construction of an SPM takes inspiration from the process of creating a knowledge graph (KG) in the semantic web [14], [42] and formulating logical clauses in ProbLog [17], [18]. Specifically, the semantic web postulates the existence of an ontology graph that outlines the structure of KGs, i.e., nodes and edges without specifying the nodal values. When real data is fed into this ontology graph, a KG appears. For each directional edge \( \rightarrow \) from a node \( A \) to \( B \), one can evaluate its truthfulness probability \( p \in [0, 1] \), which aligns with a probabilistic logic clause \( \langle p :: B:: A \rangle \) in ProbLog [17], [18]. Finally, one can quantify the uncertainty of the clause using \( p \), which coincides with semantic entropy in SIT.

Adapting this approach to MAC scenarios presents a primary challenge due to the lack of an existing ontology graph defining concepts and their connection structure. To address this problem, we treat the FP sequence of an NPM as the directional connection (i.e., causality) structure of its SPM, and derive inconsistent concepts by merging and eliminating redundant CMs. It is important to note that there are existing studies that employ MADRL or NNs for MAC designs, such as [43] for satellite networks and [44] for sensor networks. However, these works primarily focus on optimizing protocol setting parameters or access decision-makings for fixed CMs, rather than customizing CMs as done in SPMs. Meanwhile, there are KG-based semantic PHY design frameworks [45], [46], which not only overlook MAC design issues but also assume the presence of KGs without exploring how to construct KGs from real data.

E. Contributions and Organization

This work is the first of its kind to design a novel semantic MAC protocol based on logic programming and symbolic AI, extending the scope of semantic communication to network-wide applications. Like NPM, an SPM is task-specific while also retaining the symbolic and interpretable qualities found in classical MAC protocols. This approach achieves communication and memory efficiency, as well as generalization to non-stationary environments. Our main contributions can be summarized as follows:

1) **SPM Construction**: We propose a novel method to construct a ProbLog-based SPM from an NN-based NPM, which occupies only 0.02% of the NPM memory usage by extracting and merging semantically equivalent vocabularies.

2) **SPM Reconfiguration for Collision Avoidance (CA) and fairness**: Without re-training, we demonstrate that an SPM is reconfigurable for CA by identifying rules that cause collision and instantly manipulating their connections written in ProbLog. We demonstrate the extent to which reconfiguration improves the performance in a new environment, e.g. CA and fairness constrained.

3) **Best SPM Selection via Semantic Entropy**: We empirically show that minimizing the average semantic entropy of an SPM (i.e., mean uncertainty of the SPM operations) achieves the highest goodput or equivalently the highest reward, allowing one to select the best SPM in a stationary environment.

4) **SPM Portfolio for Non-Stationary Environments**: By exploiting the memory efficiency of SPMs, we propose an SPM portfolio storing a set of SPMs, each of which is the best SPM for a specific environment.

The rest of the paper is organized as follows. In Sec. II, we explain NPM from the perspective of a MAC problem and summarize its limitations. Sec. III details SPM construction based on ProbLog and shows its advantages over NPM. With extensive simulations and comparisons to several benchmarks, Sec. IV presents the significant attributes of SPM and potential applications, including CA, fairness, SPM selection, and SPM portfolio. Lastly, we conclude the work in Sec. V.

II. SYSTEM MODEL FOR PROTOCOL LEARNING

In this section, we present the system model for a MAC scenario, briefly revisit the NPM presented in [10], and discuss the potentials and limitations of NPM. Check the Appendix [10] for more details on the implementation of an NPM.

A. System Model

Suppose that the system consists of a cellular BS and multiple UEs that want to transmit their data. For convenience, it is assumed that a single unit of data transmitted by a UE, which is referred to as service data unit (SDU), can be transmitted within one communication cycle. Each UE has a total of \( D_{\text{max}} \) SDUs to transmit and SDUs are generated at a rate of \( \lambda_i \) (SDUs per cycle for UE \( i \)) until all \( D_{\text{max}} \) SDUs are generated. Each UE has a buffer of size \( b_{\text{max}} \) to store SDUs before transmissions. Thus, the buffer state of UE \( i \), denoted by \( b_i(t) \), is updated as follows:

\[
b_i(t + 1) = (b_i(t) + s_i(t) - c_i(t))^+, \quad t = 0, \ldots, T,
\]

where \( s_i(t) \) and \( c_i(t) \) represent the numbers of arrivals and departures at communication cycle \( t \), respectively,
b_i(0) = 0 for all i, T is the total number of communication cycles, and \(x^+ = \max\{0, x\}\). Here, \(c_i(t) = 1\) if UE i can successfully send one SDU in the buffer, and 0 otherwise, and \(T \geq D_{\text{max}}\).

CMs are exchanged to coordinate UEs to send their SDUs without collision. We assume that CMs are collision-free and error-free (by using low modulation level and advanced linear block error correction, e.g., [47]), while SDUs can experience a block error, which occurs at a rate of \(\epsilon\). As illustrated in Fig. 1, during a communication cycle, both the control plane and user plane communication take place between each UE and the BS. A communication cycle consists of the following 4 phases:

(i) **UCM** from UEs to BS: UEs communicate via UCMs before sending an SDU, to inform the BS about their current buffer state.

(ii) **DCM** from BS to UEs: the BS sends DCMs to the UEs to coordinate their decisions of sending the SDU or not.

(iii) **Actions**: UEs either stay silent, access the channel, or discard an SDU. First, nothing happens when a UE stays silent. Second, an SDU in the buffer is sent to the BS in a first-in-first-out (FIFO) manner, when the UE decides to access the channel. Third, the newly incoming SDU is discarded when the buffer is already full. We denote the action when UE i chooses silence (S), access (A), and discard (D), as \(a_i^S\), \(a_i^A\), and \(a_i^D\), respectively.

(iv) **ACK/NACK** signals from BS to UEs: An acknowledgement (ACK) signal is sent to notify the UE that its SDU has been successfully received. A negative ACK (NACK) signal is sent when the SDUs are lost due to collision, or due to a block error.

Meanwhile, because a cycle is shared between the control and user plane data, the communication efficiency of the CMs is related to the amount of data that can be packed into an SDU. Consequently, the main KPI is the goodput, \(n_R\), where \(n_R\) is the number of successfully received SDUs. The system objective is thus to maximize the goodput with the efficient usage of communication, memory, and computation resources.

**B. NPM Operation and Extraction**

To extract task-specific knowledge for the construction of an SPM, we start by implementing the MAC system model into an multi-agent reinforcement learning (MARL) environment as in [10], referred to as neural protocol learning. Reinforcement learning (RL) is effective for modeling the behaviors of UEs in a communication environment because it lets the UEs learn their policy emergently by rewarding desired behaviors and penalizing undesired ones, without the supervision of prelabeled data. Please check the Appendix for more details on the implementation of an NPM.

1) **NPM Operation**: The CMs and actions for UE \(i \in \{1, 2\}\) are chosen according to the following functions, which correspond to the last activation function of each NN segment as shown in Fig. 2(a) from top to bottom:

\[
\begin{align*}
\text{(NPM UCM)} & \quad u_i = g_i^U(b_i | u_i^U), \\
\text{(NPM Action)} & \quad a_i = g_i^A(d_i | w_i^A),
\end{align*}
\]

where \(u_i^U\), \(w_i^D\), \(w_i^A\) are the NN segment parameters within the NPM. According to Fig. 2(a), the following NPM operation completes a communication cycle:

(i) **UCMs**: the upper NN segment of each UE \(g_i^U\) transforms its buffer state \(b_i\) into a vector of activation values \(u_i\).

The BS concatenates the activations from the UEs into \(u = [u_1, \ u_2]\).

(ii) **DCMs**: the BS uses its NNs \(g_i^D\) and \(g_2^D\) to transform \(u\) into another activation vector \(d = [d_1, d_2]\). UE i receives \(d_i\) from the BS, \(i \in \{1, 2\}\).

(iii) **Actions**: each UE uses its bottom NN segment \(g_i^A\) to transform the vector into three activations that correspond to the Q-value of each action that the UE could take, and decide action \(a_i\) with the highest Q-value, as its action.

2) **Vocabulary and Connection Extraction**: As shown in Fig. 2, an NPM could be used for extracting vocabularies and their connections by treating it as a simulator. Then, as illustrated in Fig. 3, we can obtain an NPM extract by logically connecting all activations that correspond to input buffer states, UCMs, DCMs, and actions that are inferred simultaneously during the episodic memory, which stores the historical data on the states, intermediate activations, and actions. For convenience, denote by \(b_i^{(k)}\) a buffer state of \(b_i = k \in \{0, \ldots, b_{\text{max}}\}\). Then, a distinct UCM, denoted by \(u_i^{(k)}\), is inferred by (2). Next, for each distinct UCM pair \(u_i^{(k)}\), a distinct DCM \(d_i^{(k)}\) is inferred by (3). Lastly, an action \(a_i\) is inferred from the DCM. To cast these directed relationships as a graph, we connect the vocabularies that occur simultaneously and consequentially with a single-headed arrow. For example, if the buffer states are \(b_1^{(1)}\) and \(b_2^{(2)}\), then \(u_1^{(1)}\), \(d_1^{(2)}\), \(a_1^A\) are consequently inferred for UE 1, and \(u_2^{(2)}\), \(d_2^{(1)}\), \(a_2^A\) are inferred for UE 2. Graphing these sequential inference relationships for every input buffer state gives the NPM extract expressed as a symbolic graph in Fig. 3.

**C. Potentials and Limitations of NPMs**

In essence, an NPM is tantamount to an NN as Fig. 2(a) illustrates. In this respect, both the advantages and disadvantages of NPMs are rooted in the fundamental structures of NNs. While even a single-layer NN can represent any function [48], the current NNs are commonly overparameterized with multiple layers. Recent studies have showed that overparameterization is a key to achieving high accuracy not only in training but also in generalization, by smoothing the function, rather than finding only the exact solutions [49]. Another study has found that overparameterization makes simple first-order methods suitable for optimizing non-convex and complicated loss functions, i.e., gradient-based training, by reducing the distances among critical points, i.e., flat minima [50]. Given this, we can conclude that NPMs have great potential in learning effective CMs in given tasks. However,1

\footnote{Here, each activation value is a positive decimal number represented by a floating point 32 (FP32) data type.}
since overparameterization contributes only to the training process, most of the parameters in a trained NPM become redundant in protocol operations. This not only causes huge memory costs, but also incurs long protocol operation delays as the NN’s feed-forward latency is proportional to the number of parameters [51].

On the other hand, the CMs produced by an NPM play a role not only in protocol operations, but also as hidden activations connecting different segments of an NN. As Fig. 2(a) visualizes, the dimensions of these CM activations at UEs are smaller than the dimensions of the segment at the BS, imposing information bottlenecks during the feed-forward propagations through the NN. Since too severe bottlenecks hinder the feed-forward propagations [52], the maximum CM payload size reduction is limited inherently by the NN architecture. By experiment, we observe that in order not to significantly compromise accuracy, the CM activation dimension should be not smaller than 8 neurons. This corresponds to 32 Bytes under the 32 floating-point (FP32) arithmetic precision, which is far larger than the current control signal sizes. Furthermore, following the standard centralized training with decentralized execution (CTDE) principle in MADRL [53], CM activations from different UEs are simply concatenated at the BS. This makes the maximum number of CMs combinatorially increase with the number of UEs, which is ill-suited for 6G where the control plane architectures are envisaged to be unified under limited number of control signaling messages [1], [54].

III. SPM: SEMANTIC PROTOCOL MODEL VIA PROBABILISTIC LOGIC PROGRAMMING LANGUAGE

To obviate the shortcomings of an NPM, we propose the construction of an SPM. We make SPM communication and memory efficient by merging UCM and DCM vocabularies that exhibit redundant semantics. To greatly reduce the vocabulary sizes, we combine two merging techniques. However, combining the two leads to problematic situations where signaling vocabularies become polysemous, which we refer to as the polysemous problem. We resolve such problem by formulating SPM rules based on probabilistic logic clauses and utilizing logical inference based on contextual information stored in the empirically derived conditional probabilities.

A. An Overview of ProbLog and SPM Construction Procedure

We transform an NPM into an SPM by extracting the logical relationships from the episodic memory of MARL. To express the logical aspects of an SPM such as vocabularies, clauses, and rules, and to describe an SPM’s construction procedure succinctly and effectively, we adhere to the syntax of ProbLog [17] as follows:

(i) **Vocabularies for CMs:** We use four types of vocabularies denoted by $b_i$, $u_i$, $d_i$, $a_i$, for input buffer states, UCMs, DCMs, and actions of UE $i$, respectively. Given two vocabularies that are connected as a causal relationship, i.e. “→”, a vocabulary is referred to as the Tail if it is the cause of a causal relationship, and referred to as Head if it is the effect.

(ii) **Clauses for CM Relations** A probabilistic logic clause $c$ that connects a Head and a Tail, i.e. “with probability $p$, Head is true, if Tail is true” is expressed as $c = (p \rightarrow \text{Head}: \text{Tail})$ in ProbLog. Each vocabulary in $c$ could be accessed by $c^H = \text{Head}$ and $c^T = \text{Tail}$, and the probability is accessed by $c^p = p$, where $p$ denotes the clause’s probability of being true, i.e. the truth probability. Note that “→” expresses “−→”.

(iii) **Predicates for Semantic Clause Clustering** We use four predicates $\text{isInput}()$, $\text{isUCM}()$, $\text{isDCM}()$, and $\text{isAction}()$ that are used for the vocabularies to describe their type. An uplink clause $\alpha$ is described by $\text{isInput}(\alpha^T) = \text{isUCM}(\alpha^H) = \text{true}$, a downstream clause $\beta$ is described by $\text{isUCM}(\beta^T) = \text{isDCM}(\beta^H) = \text{true}$, and an action clause $\gamma$ is described by $\text{isDCM}(\beta^T) = \text{isAction}(\beta^H) = \text{true}$. Subsequently, we can cluster the clauses according to the clause types $\alpha$, $\beta$, and $\gamma$.

(iv) **Rules and Entailment for SPM Construction** We define a rule $R(\cdot)$ as a sequence of simultaneously occurring clauses that are logically consequent from an input. For example, given a simple protocol that consists of the two clauses $\langle A:: B \rangle$ and $\langle B:: C \rangle$, we could consequentially derive that the two clauses are true if we know that $A$ is true. We describe this relationship as entailment, denoted by $R(A) \Rightarrow B :: A$ , where $\Rightarrow$ means to logically entail.

The SPM construction requires the following procedure, which is expressed with ProbLog for clarity. First, we extract the vocabularies from the episodic memory of MARL. Second, we merge the vocabularies that exhibit redundant semantics according to two schemes. Third, we define the three logic clause types $\alpha$, $\beta$, and $\gamma$ that we can use for clustering the logical clauses according to the operation it entails such as uplink, downstream, and action. Fourth, we formulate SPM rules $R(\cdot)$ that act as the smallest element of SPM operation. Lastly, an SPM $S$ is constructed as the set of all rules that are formulated from each input buffer state.

B. SPM Construction

Fig. 4 illustrates the SPM merging process needed for constructing a communication and memory efficient SPM. The main idea is to merge the UCM and DCM vocabularies if they exhibit semantic redundancy in terms of activations and connections. The physical meaning of activations is the vector that contains the information regarding which neurons are activated, i.e. has a positive value, within the layer of interest. These vectors are then embedded as vocabularies, i.e. input states, UCMs, DCMs, and actions. The physical meaning of connections is the logical relationship between these vocabularies, which we express with clauses. This merging process results in fewer UCM and DCM bits to be transmitted for each communication cycle, and a reduction of clauses required for expressing a model. Despite its efficiency, merging CMs, which contain inter-UE semantics, introduces the cases where the DCMs become polysemous. We resolve this issue by
exploiting contextual information embedded in the empirical distributions of UCM and DCM realizations.

1) Vocabulary Extraction: We define the input state, UCM, DCM, and action vocabularies as possible outputs from the NPM generators (2)-(4), i.e. \( b_i \in \mathcal{B}_s \), \( u_i = g^v_i(b_i) \in \mathcal{U}_i \), \( d_i = g^D_i(u_i) \in \mathcal{D}_i \), and \( i \in \{1,2\} \) and \( a_i = g^\alpha_i(d_i) \in \mathcal{A}_i \) for \( i \in \{1,2\} \). First of all, the vocabularies we need to consider can be restricted to those extracted from the episodic memories within the experience replay buffer. To elaborate, assuming that the memory of the experience replay is available, we can consider the input states experienced in the memory, i.e. \( \mathcal{B}_1 \) and \( \mathcal{B}_2 \), to be the feasible domain that is plugged into (2). This is because the other states are very unlikely to be traversed. Even when the replay memory is unavailable, we can obtain one by running a few test trials. The rest of the procedure merges these vocabularies according to their semantics, and embeds them into SPM clauses that contain the semantic information regarding their relationships.

2) Activation-Aware Vocabulary Merging: As Fig. 4 illustrates, we merge the CM vocabularies by using their activation pattern, i.e. the location of non-zero elements in the activation vector extracted by \( v_i = \theta(v_i) \), where we assume ReLU activations, and \( \theta(v_i) \) is the Heaviside step function \( \theta(x) = \mathbb{I}_{x>0} \) applied to each decimal number within the vocabulary \( v_i \). For example, suppose that \( v_1 \) is the vocabulary expressed with 8 decimal numbers: \{0.382, 4.292, 0, 0, 1.249, 0, 0, 0\}. The activation pattern of \( v_1 \) is the vocabulary \{1, 1, 0, 0, 1, 0, 0, 0\}. Without activation and parameter quantization techniques [55], [56], the performance of NPMs depends heavily on the precision of the activation signals. Nevertheless, from empirical observations, we hypothesize that there is the possibility of reducing the vocabularies greatly by exploiting the activation patterns. Consequently, we merge the UCM and DCM vocabularies denoted by \( u_i^{(k_1)} \), \( u_i^{(k_2)} \), and \( d_i^{(k_1)} \), \( d_i^{(k_2)} \), respectively, for all UCMs indexed by \( k_1 \) and \( k_2 \), and DCMs indexed by \( \ell_1 \) and \( \ell_2 \), if they have the same activation pattern as follows:

\[
\begin{align*}
u_i^{(k')} &\leftarrow u_i^{(k_1)}, u_i^{(k_2)}, \quad \text{if } \theta(u_i^{(k_1)}) = \theta(u_i^{(k_2)}), \\
d_i^{(k')} &\leftarrow d_i^{(k_1)}, d_i^{(k_2)}, \quad \text{if } \theta(d_i^{(k_1)}) = \theta(d_i^{(k_2)}).
\end{align*}
\]

By using fewer vocabulary for UCM and DCM messages, the logical rules become overlapped with each other, as shown in Fig. 4(b). Because of the other UE’s impact on the decision of a DCM, the merging process makes the choice of some actions ambiguous due to polyseous DCMs, in contrast to the deterministic choices made by the NPM. To disambiguate the polyseous vocabularies, we devise a method to determine which CMs and actions have more contextual meaning for the current operation, by exploiting their likelihood represented by the clauses’ conditional probabilities.

3) Connection-Aware Merging: To further increase communication efficiency, we merge the DCMs, UCMs, and clauses altogether, identifying cases when the connected vocabularies are identical, as illustrated in Fig. 4(b). Firstly, the DCM vocabularies denoted by \( d_i^{(k_1)} \) and \( d_i^{(k_2)} \) are merged when the set of action vocabularies they are connected to are identical. The following update rule is applied for all DCMs indexed by \( \ell_1 \) and \( \ell_2 \) to merge the DCM vocabulary and its connections:

\[
d_i^{(k')} = d_i^{(k_1)}, d_i^{(k_2)}, \quad \text{if } \mathcal{A}_i^{(k_1)} = \mathcal{A}_i^{(k_2)},
\]

where \( \mathcal{A}_i^{(k)} = \{a_i \in \mathcal{A}_i \mid \beta_i = d_i^{(k)} \} \). Secondly, the UCM vocabularies denoted by \( u_i^{(k_1)} \) and \( u_i^{(k_2)} \) are merged when the set of DCMs they are connected to are identical. The following update rule is applied for all UCMs indexed by \( k_1 \) and \( k_2 \) to merge the UCM vocabulary and its connections:

\[
u_i^{(k')} = u_i^{(k_1)}, u_i^{(k_2)}, \quad \text{if } \mathcal{D}_i^{(k_1)} = \mathcal{D}_i^{(k_2)},
\]

where \( \mathcal{D}_i^{(k)} = \{d_i \in \mathcal{D}_i \mid \beta_i = u_i^{(k)} \} \). Applying connection-aware merging to UCMs reduces the UCM vocabulary further, but it makes the UCMs polysemous. Again, to resolve the polysemy problem, we embed additional semantics within the downlink clauses via the conditional probability and utilize them with the SPM rule and operation.

4) Clause Definition and Semantic Clustering: To construct an SPM clause, we need to decide the type of clauses we will construct, dictating the logical relationships of interest as follows. Technically, we could construct a clause that explains the relationship between any activation layer to any other activation layer, but we only focus on the layers corresponding to the vocabularies extracted above. We define the clause sets \( \mathcal{A}_i, \mathcal{B}_{ij}, \mathcal{F}_i \forall i, j \in \{1,2\} \) to describe the logical relationships of interest. The uplink clause set \( \mathcal{A}_i \) contains clause \( \alpha_i \), named uplink clause, which connects UE \( i \)’s input buffer state \( b_i \) to its UCM \( u_i \). The downlink clause set \( \mathcal{B}_{ij} \) contains clause \( \beta_i \), named downlink clause, which connects the UCM \( u_j \) from UE \( j \in \{1,2\} \) to the DCM \( d_i \) that will be sent to UE \( i \in \{1,2\} \).
The action clause set $\Gamma_j$ contains clause $\gamma_i$, named action clause, which connects UE $i$’s DCM $d_i$ to its action $a_i$.

To resolve polysemy, we construct SPM clauses that express the level of truth in a logical connection. The uplink, downlink, and action clauses are written in ProbLog as follows:

(Uplink clause) $\alpha_i = (1:: u_i :: b_i)$,  
(Downlink clause) $\beta_{ij} = (\Pr(d_i|u_j :: d_i :: u_j))$,  
(Action clause) $\gamma_i = (\Pr(a_i|d_i :: a_i :: d_i))$,  

where $\alpha_i \in A_i$, $\beta_{ij} \in B_{i,j}$, $\gamma_i \in \Gamma_i$, $u_i \in U_i$, $d_i \in D_i$, $a_i \in A_i$, and $i, j \in \{1, 2\}$, and the conditional probabilities are calculated empirically by simulating over the domain given by the episodic memory. The probability is 1 for (9) because $b_i$ and $u_i$ are always selected simultaneously; the conditional probability of (10) is empirically calculated as the following ratio: the number of times $u_j$ and $d_i$ are simultaneously selected divided by the total number of times $u_j$ is selected; lastly, the conditional probability of (11) is calculated as the following ratio: the number of times $d_i$ and $a_i$ are simultaneously selected divided by the total number of times $d_i$ is selected.

5) SPM Rule Formulation and SPM Construction: We construct an SPM rule, which is the basic building block of an SPM, that entails a logical sequence of clauses that connects the UCM, DCM, and action vocabularies that are causal from an input state, as illustrated in Fig. 5. Specifically, plugging in a particular $b_i$ to the NPM, we can obtain the selected vocabularies that correspond to the heads and tails of $\alpha, \beta,$ and $\gamma$ that occur simultaneously. To express this consequential relationship, we define an SPM rule for the action of UE $i$ that originates from UE $j$’s state as the following set-valued function that entails the clauses as follows:

$$R_{i,j}(b) = \{\alpha_j, \beta_{ij}^{(1)}, \ldots, \beta_{ij}^{(L)}, \gamma_i^{(1,1)}, \ldots, \gamma_i^{(L,M)}\}$$  

where the conditions for $\alpha_j, \beta_{ij}^{(\ell)}, \gamma_i^{(\ell,m)}$ entailed by $R_{i,j}(b)$ must be set for all $\ell = 1, 2, \ldots, L$ and $m = 1, \ldots, M$ as follows:

$$\alpha_j^T = b_j,$$  
$$\alpha_j^H = \beta_{ij}^{(\ell)}^T = u_j, \forall \ell = 1, 2, \ldots, L,$$  
$$\beta_{ij}^{(\ell)}^H = \gamma_i^{(\ell,m)}^T = d_i, \forall m = 1, \ldots, M,$$  
$$\gamma_i^{(\ell,m)}^H = a_i^{(m)},$$  

where $\beta_{ij}^{(\ell)} > 0$, $\gamma_i^{(\ell,m)} > 0$, and the vocabulary belonging to the same clause is highlighted with the same color. The conditions (13)-(16) describe the consequential relationship between the clauses. In (13), the state $b$ first entails a clause $\alpha_j$ that has $b$ as its tail; in (14), the UCM $u_j$ of the clause $\alpha_j$, entails all $\beta_{ij}^{(\ell)}$ that has UCM $u_j$ as its tail; in (15), the DCM of each $\beta_{ij}^{(\ell)}$ entails all $\gamma_i^{(\ell,m)}$ that has the same DCM as $\beta_{ij}^{(\ell)}$; lastly, in (16), each $\gamma_i^{(\ell,m)}$ entails an action $a_i^{(m)}$ that is implicated with a non-zero probability. An SPM is constructed as the set of all rules that originate from all input buffer state experienced in the episodic memory. Let the SPM rules be defined with (12), the SPM is constructed as follows:

$$S = \bigcup_{i,j,b} R_{i,j}(b),$$  
where $b \in B_1 \times B_2$ and $i, j \in \{1, 2\}$.

C. SPM Design Motivation and Empirical Justification

For communication and memory efficiency, we have constructed the SPM after merging the vocabularies and connections that exhibit redundant semantics. The total reduction of the vocabulary size can be seen by comparing the graph of an SPM and the blurred graph of the original NPM extracted in Fig. 6. To visualize both positive and negative effect of the merging schemes, we use the t-distributed stochastic neighbor embedding (t-SNE) [57] graphs of the UCMs and DCMs pairs, i.e. $u$ and $d$, respectively, as illustrated in Fig. 7. Fig. 7(a) shows the t-SNE graphs of vocabularies before their merging, which shows that there are multiple UCM and DCM clusters exhibiting identical semantics, expressed by identical actions pair $[a_1, a_2]$ and indicated by the same color. After activation-aware merging, we observe that UCM clusters with the same semantics are merged, but some of the DCM clusters with different semantics get semantic polysemy. On the other hand, connection-aware merging shows positive clustering of DCMs, but introduces severe polysemy for UCMs. Lastly, combining both schemes significantly merges the vocabularies further, but the polysemy occurs for both UCMs and DCMs. The results suggest that there is a minimum number of UCM and DCM vocabularies required to express the protocol without any polysemy. However, further reduction gain could be achieved if polysemous vocabularies can be disambiguated.

D. SPM Operation

To disambiguate the polysemous vocabularies, we utilize the conditional probabilities that contain contextual information regarding the environment. To express the level of confidence in the selection of a vocabulary, the truth probability of the CMs and actions for UE $i$ are calculated as follows when the entailment at state $b$ is given by $R_{i,i}(b) = \alpha_i, \beta_{ii}, \gamma_i$ and $R_{i,j}(b) = \alpha_j, \beta_{ij}, \gamma_i$:

$$\Pr(u_i|S, b) = \alpha_i^P,$$  
$$\Pr(d_i|S, b) = \beta_{ii}^{P,\beta_{ii}^P},$$  
$$\Pr(a_i|S, b) = \gamma_i^P,$$  

where $i \neq j \in \{1, 2\}$. A higher truth probability indicates higher confidence in the selection of a particular vocabulary.
Fig. 6. An example of SPM when \( N = 2 \) and \( B = 3 \), which is the transformed version of Fig. 3. The bold arrows and boxes indicate the decisions made using the maximum truth probability for input \( b_{1}^{(1)}, b_{2}^{(2)} \). The operation result makes UE 1 remain silent whereas UE 2 accesses the channel. The coloring scheme in the figure corresponds to the highlighted colors of the clauses in (9)-(11) and (24).

for the current context \( S \) and \( b \). Note that (19) is the joint probability of the event that the same DCM is implicated by the UCMs from UE \( i \) and \( j \); also, the equality for the second equation holds, because we assume that the probabilities \( \beta_{i,i}^{j} \) and \( \beta_{i,j}^{j} \) are independent. Inspired by reinforcement learning (RL) principles, which transform the soft Q-values to hard decisions, we select the CMs and actions according to the maximum truth probability as follows:

\[
\text{(SPM UCM)} \quad u_{i} = \alpha_{i}^{H}, \quad (21) \\
\text{(SPM DCM)} \quad d_{i} = \arg \max_{d \in D_{i}} \Pr(d|S, b), \quad (22) \\
\text{(SPM Action)} \quad a_{i} = \arg \max_{a \in A_{i}} \Pr(a|S, b), \quad (23)
\]

where \( i, j \in \{1, 2\} \). The advantages of using the vocabulary with the maximum truth probability are twofold: the computation cost from random sampling is saved, and the operation becomes immune to uncertainty.

Notwithstanding the amount of communication efficiency brought by the schemes explained above, it is best if actions could be taken without a grant given by the DCMs. We enable grant-free communication by comparing the SPM rules, which is referred to as rule-aware grant-free communication. When the SPM rule \( R_{i,i}(b_{i}, b_{j}) \) is unaffected by \( b_{j} \), \( j \neq i \), i.e.

\[
R_{i,i}(b_{i}, b_{j}^{(1)}) = \cdots = R_{i,i}(b_{i}, b_{j}^{(|B_{j}|)})
\]

we skip the DCM and take an action in a grant-free manner according to the fourth clause type defined as follows:

\[
\text{(Grant-free clause)} \quad \delta_{i} = \langle 1:: a_{i}; b_{i} \rangle, \quad (24)
\]

where \( a_{i} = \gamma_{i}^{H} \). Compared with our approach that enables grant-free communication with semantic vocabularies using the logical relationship, a tabular RL-based approach in [58]
uses predefined signaling messages to learn to skip the ACK signal when $\epsilon$ is low, and the deep RL-based approach in [10] learns emergent signaling messages but the DCM cannot be skipped because of the intrinsic NN structure.

Inspired by natural language based on pragmatics such as contextual meaning to disambiguate polysemous vocabularies, we have disambiguated the UCMs and DCMs via probabilistic logic-based inference in (18)-(23). The results from Sec. IV will corroborate the probabilistic logic inference’s disambiguation capability.

As a summary of the efforts in this section, Table I compares the measured KPIs of an SPM compared with an NPM, such as the goodput, vocabulary sizes, memory requirement, and so forth. The reduced CM length and model storage is consequential to the reduction gain achieved by the merging process, and the measurements in Table I are empirically derived from an actual NPM and its SPM transformation.

IV. SIMULATION RESULTS: ATTRIBUTES AND POTENTIALS

Classical protocols are typically designed for general-purpose applications, while NPMs are optimized to perform well in specific environments. Drawing inspiration from both classical protocols and NPM, an SPM possesses three key attributes that enable broader applicability: reconfigurability, measurability, and compactness. In this section, we will discuss each attribute in detail, provide examples of applications that benefit from these attributes, and present extensive experimental results to illustrate their potential.

To ensure our simulations accurately reflect realistic communication scenarios, we carefully tune communication-related hyperparameters. We set $\lambda_1 = \lambda_2 = 0.5$, $b_{\text{max}} = 5$, $D_{\text{max}} = 12$, $\rho_1 = \rho_2 = 5$, $T = 24$, and $\epsilon = 0.02$ to model a situation in which two UEs compete for a noisy channel with equal SDU arrival rates. A BS can successfully serve both UE data streams, provided that the protocol is effective. To further evaluate contention performance, we introduce additional Key Performance Indicators (KPIs), denoted by $n_C$ and $n_D$, the number of collisions and discarded SDUs, respectively.

For our implementation, we utilize the PyTorch library to build NPMs and extend the Python ProbLog library to construct SPMs. We select the Huber loss function [59] due to its robustness against outliers, which reduces the magnitude of fluctuations during NN training. We employ the Adam optimizer with an initial learning rate of 0.0001, first and second exponential decay rates of 0.9 and 0.999, and an epsilon value of $10^{-7}$ for numerical stability.

A. Comparison With NPM and Slotted ALOHA

1) SPM vs. NPM: Fig. 8 compares an NPM’s operation with that of an SPM. Fig. 8(a) plot the Q-value surfaces that correspond to the level of inclination of the UE to access the channel. Similarly, for an SPM in Fig. 8(b), the truth probability of action $a_i$ displays the level of inclination to access, because it is the value that is compared with the other actions to decide whether to access or not. The Q-values are well imitated by the SPMs’ truth probabilities for the access action in Fig. 8(a) and (b). For $b_{\text{max}} = 5$, the policy emulation is correct for 97.22% of the states. Minor deviation occurred when $b_1 = b_2 = 0$, which is acceptable because the decision when both buffers are empty is negligible, which

Fig. 8. NPM vs. SPM: the operation results of UEs and the policy maps of (a) NPM, (b) SPM. Given a buffer state, NPM returns the Q-values for each action; then, the action with the highest Q-value is selected. An SPM compares the truth probability of the actions given the input state, then chooses the action with the highest value. Plotting the decision regions for all input states gives the policy maps.
is corroborated by the identical average goodput performance when comparing S-ALOHA with random access schemes such as Slotted ALOHA (S-ALOHA) with Exponential Backoff (BEB) by 0.729 at $\lambda = 0.5$ and $\epsilon = 0.02$.

2) SPM vs. Slotted ALOHA: Fig. 9 compares the contention performance of SPM with conventional protocols. The performance of SPM is contrasted with that of random access schemes such as Slotted ALOHA (S-ALOHA) with access probability $p = 0.5$ [60] and S-ALOHA with Binary Exponential Backoff (BEB) with a base of 2 and adverse collision events. At $\lambda = 0.5$, SPM's $n_R$ surpasses S-ALOHA and S-ALOHA (BEB) by 182.2% and 206.7% respectively; and at $\epsilon = 0.01$, it exceeds them by 161.5% and 204.7%. This superior performance stems from the SPM's capacity to emulate such behavior, and logical operations that eliminate the polysemus problem. To validate performance across diverse environments, we vary packet arrival rate $\lambda$ and block error rate $\epsilon$, repeating each test 10,000 times. The performance gap between SPM and random access schemes is apparent, except when $\epsilon$ approaches 1, where all SDUs are unavoidably lost due to poor channel conditions. SPM demonstrates enhanced performance concerning the KPIs $n_C$, $n_R$, and $n_D$ across all environments. Averaging over all environments, SPM's $n_R$ exceeds that of S-ALOHA and S-ALOHA (BEB) by 142.8% and 184.4% respectively; $n_D$ is reduced to 52.7% and 47.0%; and $n_C$ is decreased to 26.4% and 50.9%.

B. Reconfigurability: Collision-Free and Fair SPM

Classical protocols can be reconfigured to address changes in the environment or system requirements, but an NPM must be retrained with the hyperparameters set according to the new constraint. This process can be demanding, as hyperparameters need reoptimization, and NPMs require many trials to converge to an optimal solution. Inspired by classical protocols, an SPM can be reconfigured to perform well for a new constraint by manipulating the logical relationships that are based on vocabularies, clauses, and rules.

1) Principles: As a principle for reconfiguration, we should make as few manipulation steps as possible to maintain the environment grounded performance of the original SPM. For example, consider reducing collisions. We can manipulate the action clause $\gamma_i \in \Gamma_i$ to reconfigure UE $i$'s action to another action. Let $\gamma_i^{(\Delta)}$ be the action clause that is entailed by $b$, which leads to a collision. We can achieve the desired reconfiguration with minimal manipulation to the SPM by updating the SPM as follows:

$$ (S, \gamma_i) \cup (\gamma_i^{P} := \gamma_i^{(\Delta)}, d_i), $$

where $R_{i,i}(b) = \alpha_i, \beta_i^{(\ell)}; \gamma_i^{(\Delta)}; \gamma_i^{H} = \gamma_i^{T} = d_i$, and the reconfiguration makes UE $i$ to stay silent instead of its original decision to access at $b$, hence avoiding collisions.

2) Reconfiguration for CA: Firstly, we show how CA performance is improved by reconfiguring an SPM. Let $n_C^{*}$ be the number of collisions that we should not exceed. For $b$, collision occurs when actions $a_1^1$ and $a_3^2$ are chosen simultaneously, which occurs with the probability $Pr(a_1^1, a_3^2 | S, b) = Pr(a_1^1 | S, b)Pr(a_3^2 | S, b)$. We decide to manipulate $\gamma$ for $b$ when $Pr(a_1^1, a_3^2 | S, b) > p_{th}$, where $p_{th}$ is the CA threshold set to satisfy $n_C^{*}$. To avoid collision for the input, we manipulate the SPM of the UE with lower access probability to remain silent instead, and choose an arbitrary UE for a tie. Suppose that UE $i$ has the lower access probability, we manipulate the $\gamma_i^{H}$ from $a_1^1$ to $a_3^2$ as explained above. Repeating this for all input states that exceeds $p_{th}$ ultimately manipulates an SPM to become collision-free. Fig. 10 shows the performance of SPM manipulated for CA. With such manipulations, $n_C$ approaches 0 for SPM after the manipulation. Only two steps...
of manipulation were required, which is much more efficient and scalable than retraining an NPM. The manipulation also resulted in a marginal improvement for $n_R$, i.e., 107.21% improvement on average for all environments, which shows that the manipulation’s impact on the SPM’s contention performance was positive.

3) Reconfiguration for Fairness: Secondly, we show how fairness could be improved by reconfiguring an SPM. Compared to ALOHA based protocols, which ensure fairness by nature, fairness is overlooked by an SPM. To ensure performance where fair utilization of the shared medium is necessary, we further manipulate the SPM by adding connections that let each UE switch their SPM with a certain probability. Notice that the general principle still holds: try to make the least manipulation as possible; this principle helps maintain the task-specific goodput performance of the original SPM. We measure the fairness by Jain’s fairness index (JFI), which is widely used for quantifying fairness in channel utilization because of its properties of scale independence and bounded value between 0 and 1 [61]. JFI is given by the following:

$$J(x_1, x_2, \ldots, x_N) = \frac{(\sum_{i=1}^{N} x_i)^2}{N \cdot \sum_{i=1}^{N} x_i^2}, \quad (26)$$

where $x_i$ is $n_R$ of user $i$ in our example. As shown in Fig. 11(a) and (b), the reconfiguration improves the fairness of SPM as well as $n_R$ between $\lambda = [0, 0.7]$, but the fairness improvement degrades the $n_R$ performance beyond that range. Moreover, as shown in Fig. 11(c), by adjusting the access probability $p$ for S-ALOHA based protocols and adjusting the SPM switch frequency, we can delineate the JFI to $n_R$ tradeoff in the protocols. The differing tradeoff curves of the protocols highlight the hidden intrinsic properties regarding fairness and contention performance of these protocols at a certain $\lambda$. In addition, the tradeoff between JFI and $n_R$ signifies that the fairness level can be controlled to optimize the $n_R$ performance while achieving the fairness constraint. The reconfiguration for achieving fairness over the entire parameter range and optimization of the fairness to $n_R$ tradeoff are left as future work.

4) Limitations: As demonstrated above, the new objectives set by a new environment can be met with reconfiguration. However, reconfiguration becomes difficult when the network topology changes. For instance, when new BSs are added to the system, to model cell-free networks [62], the BSs need to learn how to cooperate, whereas the single BS based protocol learnt by our SPM has no knowledge regarding BS cooperation, unlike the UEs. The transfer of the current semantics to another network topology is an interesting research topic, which we leave as future work.

C. Multi-SPM: Consensus in Stationary Environment and Portfolio for Non-Stationary Environment

A set of multiple SPMs can be constructed within a stationary channel environment, represented as $\{S^1, \ldots, S^{N_S}\}$, where $N_S$ denotes the number of protocols. This is due to the inherent nature of RL, which leads to non-static chan-
nel fluctuations and random realizations of SDU arrivals. Consequently, a method is required to reach a consensus on the most suitable protocol for the current environment. Drawing inspiration from the information-theoretic metrics used to assess traditional protocols, we evaluate multiple SPMs based on semantic information theory to achieve a consensus. Moreover, to address a wider range of applications, e.g. non-stationary environment, we exploit the compactness of SPMs to maintain a portfolio of SPMs.

1) Principles: The semantic information theoretic evaluation is feasible because SPMs obtain structural variability from the merging process of their respective vocabularies. On this note, we measure the level of uncertainty within a clause with the following definition of semantic entropy [45], [46]:

$$H(c) = -\{c^p \log(c^p) + (1 - c^p) \log(1 - c^p)\},$$  \hspace{1cm} (27)

where $c^p$ is the probability annotation of clause $c$, i.e. the truth probability. With equation (27), we can evaluate the net uncertainty within the SPM by taking the summation of the clause entropies:

$$H_{net}(S) = \sum_{c \in S} H(c).$$  \hspace{1cm} (28)

$H_{net}$ is insightful due to its strong correlation to the UCM and DCM vocabulary sizes $|D|$ and $|U|$, respectively, and the downlink and action clause sets’ cardinality $|B|$ and $|\Gamma|$, respectively. We can also calculate the partial net entropies for downlink clauses $\beta$ and action clauses $\gamma$ by the following: $H^\beta_{net}(S) = H_{SPM}(S \cap B)$, and $H^\gamma_{net}(S) = H_{SPM}(S \cap \Gamma)$. These metrics measure the net entropy pertaining to a specific clause type, which let us differentiate which logical connection type has more polysemous behavior within the protocol. For choosing the best fitting model, $H^\beta_{net}$ can be used by the BS to achieve a simpler control plane function, and $H^\gamma_{net}$ can be used by the UEs to lower the variance of its actions. Lastly, by utilizing the SPM entropy, we can compare the SPMs according to its SPM entropy, and select the protocol $S^\psi \in \{S^1, \ldots, S^N_S\}$ with the minimum $H_{net}(S^\psi)$ value among the protocols:

$$\arg \min_{\psi} H_{net}(S^\psi).$$  \hspace{1cm} (29)

Furthermore, we must consider a non-stationary environment to address wider range of situations. Classical protocols are designed for a general purpose, so they suffer from a non-stationary environment that deviates from the general purpose. On the other hand, NPM can be trained for a specific environment, but keeping and loading multiple models cause heavy memory overhead due to the overparametrization of NNs. In contrary, SPM’s compactness makes it easy to store and compute. For example, consider an environment where UEs have periods of bursty SDUs, UEs leave and enter, or the wireless channel deteriorates over time. To cope with these non-stationary environment, keeping multiple compact SPMs as a portfolio grant model diversity and ensemble gain [63], [64]. Thanks to the compactness of SPMs, we can maintain a portfolio of SPMs for a non-stationary environment where communication parameters change frequently.

2) SPM Consensus for Stationary Environment: Firstly, based on the aforementioned metric (28), we can quickly select one model among multiple models with simple evaluation on the protocol structure, without running performance tests. Fig. 12(a) compares the average rewards of three different selection schemes. Given a set of $N_S$ SPM samples, the random selection scheme chooses one SPM randomly from the set; min $|U| + |D|$ scheme chooses the SPM that uses the least UCM and DCM vocabularies; and the min $H_{net}$ scheme chooses the SPM with the smallest net entropy. In the experiment, we experiment with 200 SPMs, and $N_S$ SPMs are sampled from them. 3,000 trials are repeated at each $N_S$. The results show that the net SPM entropy is the best metric for getting a consensual SPM among the three, because it selects the well-performing SPM from smaller sample sizes. Selecting according to the minimum net entropy gives an average reward that converges to 3.84 around $N_S = 20$ with the standard deviation of 0.22, and choosing for the minimum vocabulary size gives an average reward that converges to 3.84 around $N_S = 60$, whereas random selection scheme’s average reward stays between 1.5 and 1.8 for all sample sizes, with the average standard deviation of 1.86.

3) SPM Portfolio for Non-Stationary Environment: To verify the robustness of an SPM portfolio, we compare it with a continual learning based NPM over a changing environment. Fig. 12(b) compares two schemes that try to adapt to a
frequently changing environment, and an NPM that is trained for a stationary environment, to provide a baseline. In the simulation, we try to model a non-stationary environment with UEs that have incoming bursts of SDUs by a two-state Markov chain, where in one state UE 1 has $\lambda = 0.9$ and UE 2 has $\lambda = 0.1$, and in the other state, the packet arrival rates are reversed; the state transition probability is 0.8 for both states. In such a dynamic environment, a protocol that strongly prioritizes only one UE to transmit suffers when the environment state transition happens. The red triangle graph shows how the training would have progressed if the environment was stationary, with each UE having $\lambda = 0.5$; the blue circled graph shows the test performance of a protocol being adjusted by continual learning, which retrains at the current environment, starting from the NPM of the previous state. The purple diamond graph shows the test performance of an SPM portfolio, with SPMs constructed for diverse environments. The figure shows that continual learning leads to detrimental results, i.e. negative average reward for 74.3% of the communication cycles, which is due to catastrophic forgetting [65], and learning stuck at bad local minima. In contrast, an SPM portfolio never falls below the average reward of 2.63 because the portfolio contains a model that is constructed for each environment.

V. Conclusion

In this work, we proposed a novel semantic MAC protocol model extracted and symbolized from an NN-based NPM, coined an SPM. The SPM shifts the paradigm of semantic communication from point-to-point semantics to network-wide semantics by transforming the environment grounded black-box NN into an interpretable collection of semantic clauses written in ProbLog, a logic programming language for symbolic AI. The SPM is therefore interpretable by both humans and machines, solidly grounded on the communication environment, and instantly reconfigurable to fit to new communication environments such as collision and fairness constrained environments. Furthermore, we can measure the semantic entropy of an SPM, which compactly occupies only 0.02% of the memory compared to its NPM counterpart, to select the best SPM and maintain SPM portfolios. There are several intriguing future problems such as extending to dynamic spectrum access, viewing this problem as an NPM that is constructed for diverse environments. The figure shows that continual learning leads to detrimental results, i.e. negative average reward for 74.3% of the communication cycles, which is due to catastrophic forgetting [65], and learning stuck at bad local minima. In contrast, an SPM portfolio never falls below the average reward of 2.63 because the portfolio contains a model that is constructed for each environment.

APPENDIX

In the Appendix, we explain the details regarding the implementation of MARL for modeling a MAC environment, and walk through how an NPM is learned and how its environment is set up. First of all, the problem is formulated as a decentralized partially observable Markov decision process (Dec-POMDP) [66] with a communication channel between the UEs and the BS. Consequently, NPM follows centralized training with decentralized execution (CTDE) to address non-stationarity [10]. A multi-agent deep-Q-network (DQN) is chosen in particular to construct an NPM, because it leverages NNs and experiences replay to approximate the Q-function.

States, Actions, Observations, and Rewards: In order to see how agents interact closely, we assume that there are two UEs. For convenience, UEs and BS, which are assumed to be decentralized agents, are indexed by $i \in \{1, 2, 3\}, i = 3$ being assigned to the BS. Agents make partial observations on their current state by observing their buffer states:

$$b = [b_1, b_2]$$

where $b_i \in \{0, 1, 2, \ldots, b_{\text{max}}\}$ and $i \in \{1, 2\}$. Depending on $b$ and the current protocol, UE $i$ can choose its action as follows:

$$a_i = \begin{cases} 
S, & \text{silence}, \\
A, & \text{access the channel and send the oldest SDU}, \\
D, & \text{discard the oldest SDU from the buffer}. 
\end{cases}$$

(30)

For notational clarity, we denote the action when UE $i$ chooses $S$, $A$, and $D$, as $a_i^S$, $a_i^A$, and $a_i^D$, respectively. When UE $i$ selects $a_i^A$, it is assumed that $c_i = 1$. In this work, techniques are targeted towards resource-constrained IoT devices, and it is implicitly assumed that no re-transmission is allowed to avoid excessive resource consumption [67]. It is interesting to extend the current design towards incorporating re-transmission schemes, e.g., the approach in [68], which is outside the scope of the current work. According to the contention and decoding results, one of the following observations is available at the BS:

$$o = \begin{cases} 
\text{idle}, & \text{no SDU is received}, \\
\text{ACK}_i, & \text{received and decoded SDU from UE } i, \\
\text{NACK}, & \text{failed to receive or decode an SDU}. 
\end{cases}$$

(31)

According to the BS’s observation and actions taken by the UEs, a reward is given to the agents by a central critic. The rewards are defined as follows:

$$r = \begin{cases} 
+\rho_1, & \text{if one SDU packet is successfully received}, \\
-\rho_2, & \text{if a UE discards an SDU packet from its buffer}, \\
-1, & \text{otherwise}. 
\end{cases}$$

(32)

where $\rho_1, \rho_2 > 0$ are the design parameters.

NPM Learning: NPM aims to maximize the average system reward by approximating the optimal Q-function for a given environment. The Q-function is the expectation of the sum of discounted future rewards [69]. To construct an NPM for such a scenario, we optimize the DQN weight parameters in a centralized manner, with a central critic calculating the reward. As shown in Fig. 13, the hyperparameters are set to achieve a high average reward. Note that more recent architectures and training methods such as self-attention based multi-agent architecture [70] and random distillation based exploration [71] could be used complementarily to further improve the performance. However, for this work, the effect
hidden layer and 16 nodes are used for each NN segment for the BS. Each NN segment consists of two hidden layers, and the output layer pertaining to a CM or the Q-values. The layers are activated with ReLU, and the hyperparameters are tuned to maximize the average rewards.

was not significant enough to warrant further complicating the learning process, which we leave as future work. As illustrated in Fig. 2(a), the NPM architecture consists of two NN segments for each UE (i.e., a total of 4 NN segments for two UEs), and one NN segment for the BS. Each NN segment consists of two hidden layers, and the output layer pertaining to a CM or the Q-values. The layers are activated with ReLU, and the hyperparameters are tuned to maximize the average goodput. For the NN segments, 16 nodes are used for each hidden layer and 8 nodes are used for the outputs.

Fig. 13. NPM needs (a) CM length of at least 16 output activation nodes, and (b) 16 intermediate activation nodes to ensure convergence to optimal average rewards.

REFERENCES

[1] S. Wang, T. Sun, H. Yang, X. Duan, and L. Lu, “6G network: Towards a distributed and autonomous system,” in Proc. 2nd 6G Wireless Summit, Mar. 2020, pp. 1–5.

[2] D. Gündüz et al., “Beyond transmitting bits: Context, semantics, and task-oriented communications,” IEEE J. Sel. Areas Commun., vol. 41, no. 1, pp. 5–41, Jan. 2023.

[3] X. Luo, H. Chen, and Q. Guo, “Semantic communications: Overview, open issues, and future research directions,” IEEE Wireless Commun., vol. 29, no. 1, pp. 210–219, Feb. 2022.

[4] Medium Access Control (MAC) Protocol Specification, 3GPP, document TS 38.321, 2020.

[5] K. Chen, M. Ma, E. Cheng, F. Yuan, and W. Su, “A survey on MAC protocols for underwater wireless sensor networks,” IEEE Commun. Surveys Tuts., vol. 16, no. 3, pp. 1433–1447, 3rd Quart., 2014.

[6] M. B. Shahab, R. Abbas, M. ShirvaniMoghaddam, and S. J. Johnson, “Grant-free non-orthogonal multiple access for IoT: A survey,” IEEE Commun. Surveys Tuts., vol. 22, no. 3, pp. 1805–1838, 3rd Quart., 2020.

[7] M. Giordani and M. Zorzi, “Non-terrestrial networks in the 6G era: Challenges and opportunities,” IEEE Netw., vol. 35, no. 2, pp. 244–251, Mar. 2021.

[8] M. Noor-A-Rahim et al., “6G for vehicle-to-everything (V2X) communications: Enabling technologies, challenges, and opportunities,” Proc. IEEE, vol. 110, no. 6, pp. 712–734, Jun. 2022.

[9] M. Maier, A. Ebrahizadeh, S. Rostami, and A. Beniiche, “The Internet of No Things: Making the internet disappear and ‘see the invisible,’” IEEE Commun. Mag., vol. 58, no. 11, pp. 76–82, Nov. 2020.

[10] M. P. Mota, A. Valcarce, J.-M. Gorce, and J. Hoydis, “The emergence of wireless MAC protocols with multi-agent reinforcement learning,” 2021, arXiv:2108.07144.

[11] J. Hoydis, F. A. Aoudia, A. Valcarce, and H. Viswanathan, “Toward a 6G AI-native air interface,” IEEE Commun. Mag., vol. 59, no. 5, pp. 76–81, May 2021.

[12] S. Han et al., “Artificial-intelligence-enabled air interface for 6G: Solutions, challenges, and standardization impacts,” IEEE Commun. Mag., vol. 58, no. 10, pp. 73–79, Oct. 2020.

[13] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, “Learning to communicate with deep multi-agent reinforcement learning,” in Proc. Adv. Neural Inf. Process. Syst., vol. 29, 2016, pp. 1–13.

[14] T. Berners-Lee, J. Hendler, and O. Lassila, “The semantic web,” Sci. Amer., vol. 284, no. 5, pp. 34–43, 2001.

[15] D. E. Klein and G. L. Murphy, “The representation of polysemous words,” J. Memory Lang., vol. 45, no. 2, pp. 259–282, Aug. 2001.

[16] E. Huang, R. Socher, C. Manning, and A. Ng, “Improving word representations via global context and multiple word prototypes,” in Proc. 50th Annu. Meeting Assoc. Comput. Linguistics, vol. 1, 2012, pp. 873–882.

[17] H. T. Luc De Raedt and A. Kimmig, “ProbLog: A probabilistic prolog and its application in link discovery,” in Proc. IJCAI, Jan. 2007, pp. 2468–2473.

[18] L. D. Raedt, S. Dumančić, R. Manhaeve, and G. Marra, “From statistical relational to neuro-symbolic artificial intelligence,” in Proc. 29th Int. Joint Conf. Artif. Intell., Jul. 2020, pp. 1–15.

[19] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, “Wireless network intelligence at the edge,” Proc. IEEE, vol. 107, no. 11, pp. 2204–2239, Nov. 2019.

[20] J. Park et al., “Communication-efficient and distributed learning over wireless networks: Principles and applications,” Proc. IEEE, vol. 109, no. 5, pp. 796–819, May 2021.

[21] C. E. Shannon and W. Weaver, The Mathematical Theory of Communication, vol. 117. Champaign, IL, USA: Univ. Illinois Press, 1949.

[22] C. E. Shannon, “A mathematical theory of communication,” Bell Syst. Tech. J., vol. 27, no. 3, pp. 379–423, Jul. 1948.

[23] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, “Sparse representation for computer vision and pattern recognition,” Proc. IEEE, vol. 98, no. 6, pp. 1031–1044, Jun. 2010.

[24] P. A. Stavrou and M. Kountouris, “A rate distortion approach to goal-oriented communication,” in Proc. IEEE Int. Symp. Inf. Theory (ISIT), Jun. 2022, pp. 590–595.

[25] H. Xie, Z. Qin, G. Y. Li, and B. Juang, “Deep learning enabled semantic communication systems,” IEEE Trans. Signal Process., vol. 69, pp. 2663–2675, 2021.

[26] E. Bourtsoulatzis, D. Burth Kurka, and D. Gündüz, “Deep joint source-channel coding for wireless image transmission,” IEEE Trans. Cognit. Commun. Netw., vol. 5, no. 3, pp. 567–579, Sep. 2019.

[27] L. Yan, Z. Qin, R. Zhang, Y. Li, and G. Y. Li, “Resource allocation for text semantic communications,” IEEE Wireless Commun. Lett., vol. 11, no. 7, pp. 1394–1398, Jul. 2022.

[28] Z. Weng, Z. Qin, and G. Y. Li, “Semantic communications for speech signals,” in Proc. IEEE Int. Conf. Commun., Jun. 2021, pp. 1–6.

[29] D. Huang, F. Gao, X. Tao, Q. Du, and J. Lu, “Toward semantic communications: Deep learning-based image semantic coding,” IEEE J. Sel. Areas Commun., vol. 41, no. 1, pp. 55–71, Jan. 2023.

[30] X. Luo, R. Gao, H. Chen, S. Chen, Q. Guo, and P. N. Suganthan, “Multi-modal and multi-user semantic communications for channel-level information fusion,” IEEE Wireless Commun., early access, Oct. 17, 2022, doi: 10.1109/MWC.011.2200288.

[31] A. Li, X. Wei, D. Wu, and L. Zhou, “Cross-modal semantic communications,” IEEE Wireless Commun., vol. 29, no. 6, pp. 144–151, Dec. 2022.

[32] M. Weber, C. Renggli, H. Grabner, and C. Zhang, “Observer dependent lossy image compression,” 2019, arXiv:1910.03472.

[33] A. M. Girgis, J. Park, C. Liu, and M. Bennis, “Predictive control and communication co-design: A Gaussian process regression approach,” in Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC), May 2020, pp. 1–5.
Seung-Woo Ko (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the School of Electrical and Electronic Engineering, Yonsei University, South Korea, in 2006, 2007, and 2013, respectively. Since 2022, he has been an Associate Professor with the Department of Smart Mobility Engineering, Inha University. Before joining Inha University, he was an Assistant Professor with Korea Maritime and Ocean University (KMOU), Busan, South Korea from 2019 to 2022, a Senior Researcher with LG Electronics, South Korea, from 2013 to 2014, and a Post-Doctoral Researcher with Yonsei University, South Korea, from 2014 to 2016, and The University of Hong Kong (HKU) from 2016 to 2019. His research interests include intelligent wireless communications and networking for 6G, with special emphasis on semantic communications, V2X communications and positioning, and radio-based localization.

Jinho Choi (Fellow, IEEE) was born in Seoul, South Korea. He received the B.E. degree (magna cum laude) in electronics engineering from Sogang University, Seoul, in 1989, and the M.S.E. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 1991 and 1994, respectively. He is currently with the School of Information Technology, Deakin University, Australia, as a Professor. Prior to joining Deakin University in 2018, he was with Swansea University, U.K., as a Professor/the Chair in wireless, and the Gwangju Institute of Science and Technology (GIST), South Korea, as a Professor. He has authored two books published by Cambridge University Press in 2006 and 2010 and one book by Wiley-IEEE in 2022. His research interests include the Internet of Things (IoT), wireless communications, and statistical signal processing. He received a number of best paper awards, including the 1999 Best Paper Award for Signal Processing from EURASIP. He has been on the list of World’s Top 2% Scientists by Stanford University since 2020. He is currently a Senior Editor of IEEE WIRELESS COMMUNICATIONS LETTERS and a Division Editor of Journal of Communications and Networks (JCN). He has also served as an Associate Editor or an Editor for other journals, including IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE COMMUNICATIONS LETTERS, JCN, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and ETRI Journal.

Mehdi Bennis (Fellow, IEEE) is currently a Full (tenured) Professor with the Centre for Wireless Communications, University of Oulu, Finland, and the Head of the Intelligent Connectivity and Networks/Systems Group (ICON). He has published more than 200 research papers in international conferences, journals, and book chapters. His main research interests are in radio resource management, game theory, and distributed AI in 5G/6G networks. He was a recipient of several prestigious awards, including the 2015 Fred W. Ellersick Prize from the IEEE Communications Society, the 2016 Best Tutorial Prize from the IEEE Communications Society, the 2017 EURASIP Best Paper Award for the Journal of Wireless Communications and Network, the All-University of Oulu Award for research, the 2019 IEEE ComSOC Radio Communications Committee Early Achievement Award, and the 2020 Clarivate Highly Cited Researcher by the Web of Science. He is an Editor of IEEE TRANSACTIONS ON COMMUNICATIONS and a Specialty Chief Editor of Data Science for Communications and the Frontiers in Communications and Networks journal.

Seong-Lyun Kim received the B.S. degree in economics from Seoul National University, Seoul, South Korea, and the M.S. and Ph.D. degrees in operations research (with application to wireless networks) from the Korea Advanced Institute of Science and Technology. He was an Assistant/Associate Professor of radio communication systems with the Department of Signals, Sensors and Systems, Royal Institute of Technology (KTH), Stockholm, Sweden. He was also a Visiting Professor with the Control Engineering Group, Helsinki University of Technology (now Aalto), Finland, the KTH Center for Wireless Systems, and the Graduate School of Informatics, Kyoto University, Japan. He is currently a Professor and the Former Head of the School of Electrical and Electronic Engineering, Yonsei University, Seoul, heading the Robotic and Mobile Networks Laboratory (RAMO) and the Center for Flexible Radio (CFR+). He recently co-directed the H2020 EUK PriMO-5G Project, and led the Smart Factory Committee of 5G Forum, South Korea. He consulted various companies in the area of wireless systems both in South Korea and abroad. He has published numerous papers, including the coauthored book Radio Resource Management for Wireless Networks (with Prof. Jens Zander). His research interests include radio resource management, AI/ML in wireless networks, collective intelligence, and robotic networks. He served as a technical committee member or a chair for various conferences, and an Editorial Board Member for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS LETTERS, Control Engineering Practice (Elsevier), ICT Express (Elsevier), and Journal of Communications and Networks. He served as the Leading Guest Editor for IEEE WIRELESS COMMUNICATIONS and IEEE NETWORK for wireless communications in networked robotics, and IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. He was a co-recipient of the IEEE VTC Best Paper Award, the IEEE Dyspan Best Demo Award, and the IEEE Heinrich Hertz Award.