按需界面：面向 6G 的原生 AI 控制接口

Abhishek Dandekar
TU Berlin
Berlin, Germany
a.dandekar@tu-berlin.de

Prashiddha D. Thapa
Fraunhofer HHI
Berlin, Germany
prashiddha.dhoj.thapa@hhi.fraunhofer.de

Ashrafur Rahman
TU Berlin
Berlin, Germany
a.rahman@tu-berlin.de

Julius Schulz-Zander
Fraunhofer HHI
Berlin, Germany
julius.schulz-zander@hhi.fraunhofer.de

摘要—传统的标准化网络接口面临显著的限制，包括供应商特定的不兼容性、僵化的设计假设以及缺乏适应新功能的能力。我们提出了一种多代理框架，利用大型语言模型（LLMs）按需生成网络功能（NFs）之间的控制接口。这包括一个匹配代理，用于将所需的控制功能与 NF 能力对齐，以及一个代码生成代理，用于生成实现接口所需的 API 服务器。我们在模拟的多供应商 gNB 和 WLAN AP 环境中验证了我们的方法。性能评估突出了在界面生成任务中 LLMs 之间成本与延迟之间的权衡。我们的工作为 AI 原生动态控制接口生成奠定了基础，为未来的移动网络增强了互操作性和适应性。

***Index Terms*—人工智能原生，接口，6G，代理，大语言模型**

I. 介绍

移动网络随着时间的推移，从单一且紧密耦合的状态演变为更加灵活的形式。这一演变始于软件定义网络（SDN）的出现，它将网络控制平面与数据平面分离开来。这意味着不再需要紧密耦合的软硬件，可以将软件独立于硬件远程运行。这为分散式网络铺平了道路。在分散式网络中，不是在一个单一的网络功能（NF）中组合所有功能，而是将其分割成多个 NF。例如，在 5G 网络中，一个单独的 gNB 可以被拆分为无线单元（RU）、分布式单元（DU）、控制单元-控制平面（CU-CP）和控制单元-用户平面（CU-UP）[1]。当单一的 NF 被分割为多个 NF 时，它们需要新的接口来彼此通信。这些接口可能是控制接口（例如 F1-C）或用户数据接口（例如 F1-U）。整个移动网络可以被视为一组通过这些接口互相通信和协调的 NF。

为了操作具有分散和分布式网络功能的网络，必须提前定义这些网络接口。这是由各种标准化组织（SDOs）完成的。然而，使用这种预定义的标准接口也会产生各种问题。

- 所有 NF 的供应商可能无法符合这些标准化接口，从而使得多供应商部署变得具有挑战性。
- 标准化的网络接口通常特定于某种无线电接入技术（RAT），这意味着属于一种 RAT 的 NF 无法与另一种 RAT 通信。
- 这些接口不够灵活，并且假定 NF 的功能保持不变而设计。如果要向 NF 添加新功能，则无法使用预定义的网络接口。

为了应对这些挑战，我们提出了一种框架，在该框架中我们可以按需在任意两个 NF 之间创建控制接口。我们通过基于大型语言模型（LLMs）的多代理框架实现了这一点。

在第二节中，我们简要介绍了大语言模型的背景，接着在第三节概述了我们的框架。第四节探讨了使用我们框架实现的一个用例，第五节描述了其实施和评估。在第六节中，我们讨论了我们框架当前的局限性。在第七节中，我们提供了相关研究工作，并在最后一节总结了我们的工作。

II. 背景

A. 大型语言模型

大型语言模型是基于大量文本数据训练的机器学习模型，用于生成自然语言文本 [2]。这些模型本质上是非确定性的，并通过预测序列中的下一个单词来操作，逐步构建连贯且与上下文相关的文本。为了生成输出，需要为这些模型提供一组称为提示的指令。上下文窗口指的是大型语言模型一次可以处理和生成的文本或数据量。由于大型语言模型具有有限的上下文窗口，因此可以在提示中生成和提供的文本量是有限的。在需要提供大量文本的情况下，会使用诸如检索增强生成（RAG）[3] 等技术。

使用 RAG，大语言模型利用检索到的信息生成既基于事实又高度相关且准确的输出。这种方法显著减少了产生不正确或无关回答的机会，并提供了额外的好处，如跨领域的增强适应性、最新数据的实时集成以及通过引用来源提高用户信任度。RAG 确保最小化幻觉现象并生成精确、可靠和定制化的输出。

B. 代理 *LLMs*

一个代理 [4] 可以被视为建立在大语言模型引擎之上的软件管道集合，允许访问外部信息、交叉检查和推理。之所以称为代理是因为它具有机构，即无需人类持续输入即可自主行动的能力。一个代理可以专门执行特定任务。在一个多代理系统中，多个这样的专门代理协同工作以实现某个目标。例如，行程代理可以根据用户需求创建旅行计划。然后该代理可以将行程转发给机票预订代理，后者可以从各个航空公司获取信息并预订最合适的机票。此类多代理框架可用于从一组需求达到某个目标。

III. 接口生成框架

本节概述了生成两个 NF 之间控制接口所需的关键步骤。

考虑两个 NF 集合，一个源 NF（源 *NF*）和一个目标 NF（*NF* 目标地），如图 1 所示。假设 *NFsrc* 需要生成一个控制接口面向 *NF* 目标地以控制其功能。它使用两个基于 LLM 的代理，匹配代理在 *NF* 源中和代码生成代理在 *NF* 目标中来生成这个接口。在接口生成过程开始之前，*NF* 源需要使用适当的方法（例如基于 DNS 的发现、静态配置等）来发现 *NF* 目标地。随后，它需要通过合适的机制与 *NF* 目标建立信任关系。然后在 *NFsrc* 和 *NF* 目标之间基于预定义的端口建立一个供应接口。该接口由相应的代理用于相互通信。

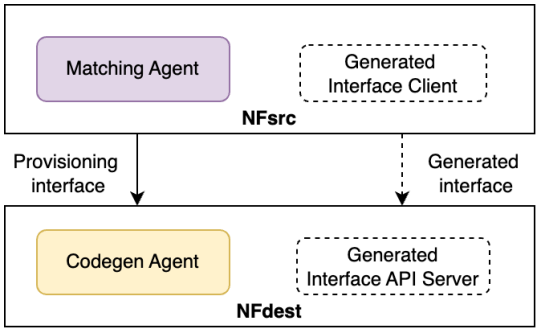

图 1. 多智能体框架

A. 匹配代理

生成接口的初始步骤是满足两种类型的兼容性要求。首先，两个 NF 都需要在功能上兼容，这意味着目标 *NF* 目标应该具有支持由源节点故障所需的功能的能力（例如更改传输通道）。其次，它们需要在语义上兼容，这意味着 *NF* 目标需要能够正确理解来自 *NFsrc* 的传入控制消息，而不管函数名称的变化。例如，*NF* 目标需要理解设置通道和设置 *chn* 都指的是设置信道号码的同一功能。两个 NF 还需要为控制消息（例如协议缓冲区、平板缓冲区等）具有兼容的编码/解码方案。

当用户打算将某些控制功能从 *NF* 源发送到 *NF* 目标时，需要一种机制来检查所需的功能是否由 *NF* 目标地支持。为了实现这一点，用户向匹配代理发送控制功能需求（图 2 步骤 1）。匹配代理是一个基于大型语言模型的代理，允许用户查找所需的控制功能是否得到支持，并识别所需的输入参数。这是通过将用户的所需控制功能描述与 *NF* 目标能力文档进行匹配来实现的（图 2 步骤 2）。*NF* 目标地能力文档详细说明了 *NF* 目标的能力。它可以来自一个在线的能力文档仓库，也可以直接来自 *NF* 目标自身。

如果在 NF 控制功能中找不到完全匹配项，则选择最接近的匹配项，该匹配项可以由 codegen 代理进行增强的。如果代理无法找到任何相似的功能，它将通知用户 NF 无法支持所需控制功能。请注意，这种匹配是根据可用功能进行的，而不是根据 *NF* 目标的确切 API 进行的。这允许 NF 供应商不对外公开其 API。

如果 *NF* 目标支持所需的功能，则会创建一个控制功能需求（CFR）文档。该文档包含所需的控制函数列表及其与能力文档中匹配的函数，以及所需的编码方案。它可以包括所需的控制函数名称、输入参数、输出参数、控制函数描述和匹配的控制函数。还包括参数数据类型、单位等详细信息。例如：

func setpower (radioID string, pow string dBm)(response boolean): <描述 >:<匹配函数>

匹配代理随后生成一个接口客户端，该客户端允许发送和接收控制功能消息至 *NF* 目标（图 2 步骤 3）。然后将 CFR 文档发送到 *NF* 目标地（图 2 步骤 4）。可以通过配置接口上的 REST POST 查询来发送此文档。

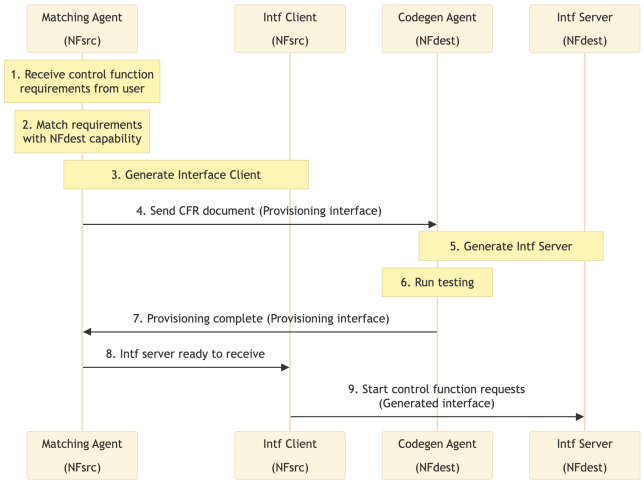

图 2. 界面生成工作流

B. 代码生成代理

来自匹配代理的 CFR 文档被 codegen 代理在 *NF* 目标接收到。此代理的任务是创建一个接口 API 服务器应用程序，可以解码并执行 CFR 文档中提供的控制功能。codegen 代理使用存在于 *NF* 目标中的内部 API 文档的知识生成此接口 API 服务器（图 2 第 5 步）。内部 API 文档是特定于供应商的开发者文档，详细说明了如何使用内部 API 触发由 *NF* 目标地支持的控制功能。内部 API 和 API 文档可以专有，并且不需要对外公开。生成的接口 API 服务器执行以下功能：

- 解码由接口客户端发送的来自 *NFsrc* 的控制消息
- 如果需要，请调整控制函数参数的单位或数据类型
- 触发相应的内部 API
- 如果需要，请调整从内部 API 返回的参数的单位和数据类型
- 编码并发送此信息回 *NFsrc* 接口客户端

除了这一点，代码生成代理还可以生成增强功能。这发生在所需的控制函数在内部 API 中没有直接匹配的函数时。代码生成代理围绕与控制函数所需额外功能最相似的内部 API 函数生成一个包装器函数。

一旦代码生成完成，代理将生成一个带有虚拟参数值的测试接口客户端代码以检查生成的接口服务器的有效性。此测试代码也可能由*源_nfsrc* 提供。如果接口服务器代码在测试中抛出错误，则会将该错误反馈给 LLM 以重新生成此代码。代理可以多次执行此操作，直到生成正确的代码版本（图 2 第 6 步）。一旦生成了正确的代码，代码生成代理向*源节点故障*发送一个配置完成的消息，表示新的控制接口已准备好（图 2 第 7 步）。然后匹配代理触发接口客户端以启动连接（图 2 第 8 步）。在此步骤之后，接口客户端与接口服务器建立连接并开始发送控制函数请求（图 2 第 9 步）。

IV. 用例

在本节中，我们提供了一个示例，说明如何使用前几节描述的多代理框架按需生成用于控制 IEEE 802.11（WLAN）和 5G NFs 的接口。O-RAN 是一个将 5G gNB-DU 解聚为 O-DU 和 O-RU 的框架。根据 SDN 的原则，它还将控制平面从 5G gNB 中分离出来，并将其放置在两个不同的控制器 [5] 中。非实时 RIC（Non-RT RIC）控制具有较高延迟的任务（>1s ），而准实时 RIC（Near-RT RIC）处理需要较低延迟的控制任务（1 秒<）。非实时 RIC 使用 O1 控制接口向 DU 和 CU 发送控制消息，而准实时 RIC 则通过 E2 控制接口控制这些 NFs。这两个接口均由 O-RAN 联盟标准化。然而，它们存在以下问题：

- 部署多供应商的 O-RAN 具有挑战性。尽管 DU/CU 可能使用标准化的 E2 接口，但它可能与近实时 RIC 不兼容。这是由于实现上的分歧，尤其是在编码方案方面。一个很好的例子是 OSC-RIC [6] 和 FlexRIC [7]。由于它们有不同的 E2 实现，每个 DU/CU 都需要有一个 OSC-RIC E2 代理和一个 FlexRIC E2 代理以便与近实时 RIC 通信。简而言

之，控制器和 DU/CU 在功能上是兼容的，但在语义上不兼容。

- 对于像 E2 这样的预定义标准化接口进行更改需要修改服务模块（SM）以及 DU/CU 的底层代码。这限制了创新，因为它减慢了开发周期。
- 3GPP 允许使用非 3GPP 技术，如 WLAN 与 5G 结合。然而，像 O-RAN 这样的 SDOs 没有指定面向 WLAN 的控制接口，因此无法实现 3GPP 5G 和 WLAN 的联合控制。这些网络目前不会相互适应以达到最优资源利用。为了实现联合控制，我们需要一个多 RAT RIC，它可以同时控制 gNB 和 AP。通过多 RAT RIC，RIC 应用可以协调并控制 5G 和 WLAN 网络 [8]。例如，如果 WLAN 链路提供更可靠的延迟，而 5G 链路提供更高的吞吐量，则可以根据需要调整资源分配以优先考虑 WLAN 链路上的延迟和 5G 链路上的吞吐量。

图 3 展示了 5G 和 WLAN 联合部署的一个场景。请注意，可以有多个来自不同供应商的 AP 和 gNB。WLAN 通过 3GPP 定义的 N3IWF（N3 互操作功能）与 5G 核心网络通信作为 [1]。核心网络将 WLAN 接入点 (AP) 视为透明，并不对它进行控制。为了控制 WLAN AP，多 RAT RIC 生成 G2，同时为控制 gNB 生成 G1。

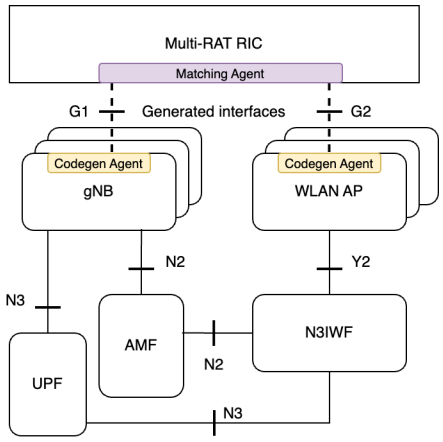

图 3. WLAN 和 5G 的联合控制

最初，RIC 发现并建立与所有目标 gNB 和 AP 的受信任连接。用户然后向匹配代理提供控制功能需求。该代理随后生成带有匹配的 gNB 或 AP 控制功能的 CFR 文档。此 CFR 文档随后发送到 gNB 和 AP。这些信息由每个 NF 上的代码生成代理接收。这些代理生成创建接口 API 服务器所需的代码。请注意，API 服务器对每个供应商都是唯一的。该代理测试代码直到生成正确的版本。一旦成功，它会向 RIC 发送一个配置完成消息。随后，RIC 触发接口客户端开始发送控制消息。

V. 实现与性能

为了测试我们的用例，我们使用了一个带有嵌入式匹配代理的自定义多 RAT RIC 实现，如图 3 所示。我们使用容器模拟 gNB 和 WLAN 接入点。由于商用供应商不会公开其内部 API，我们为 gNB 和 AP 创建了一组自定义内部 API，可以触发模拟控制功能。例如，*获取速率统计和释放 UE*。我们的 API 分别模拟了 30 个针对 gNB 和 AP 的控制动作。为了模拟多供应商场景，我们创建了多个功能相似但语义不同的 API 集（例如，变化的功能名称、数据类型和单位）。我们使用 5 个模拟的多供应商 gNB 和 5 个模拟的多供应商接入点进行测试。

对于两个代理，我们使用结合了两种大语言模型的 RAG 管道：GPT-4o 和 Llama3.3-70B。GPT-4o 模型在 OpenAI 服务器上运行，而 Llama3.3 模型则在一个配备 Nvidia Tesla V100（128GB 显存）的服务器上运行。我们使用 bge-small-en-v1.5 作为与 Llama3.3 配合的嵌入模型，并将 text-embedding-ada-002 用作 GPT-4o 的嵌入模型。

RAG 管道由三个阶段组成——文档索引、基于查询的检索和响应生成。在文档索引阶段，输入文本文档被标记化以创建嵌入。这些嵌入存储在向量数据库中以便以后检索。在基于查询的检索阶段，搜索向量数据库以找到与 LLM 输入查询最相关的文本嵌入。获取前三个最相关的嵌入并传递给 LLM。在响应生成阶段，LLM 根据检索到的文本嵌入为输入查询生成回复。对于匹配代理而言，NF 能力文档和用户的控制功能要求被用作 RAG 管道的输入文档。对于代码生成代理，CFR 文档以及供应商 API 文档被用作其 RAG 管道的输入。我们使用经过专门调整以适应这些代理功能的自定义提示。

A. 匹配代理性能

为了测试匹配代理的性能，我们根据 AP/gNB 的能力创建了 60 个自定义控制功能需求。为了考虑用户输入要求的多样性，我们使用 Claude 3.5 Sonnet [9] 生成了这些要求的十个不同变体。然后我们将这些变体作为输入提供给匹配代理，并观察它是否能正确地将它们与 AP/gNB 的能力相匹配以及需要多少次尝试才能实现正

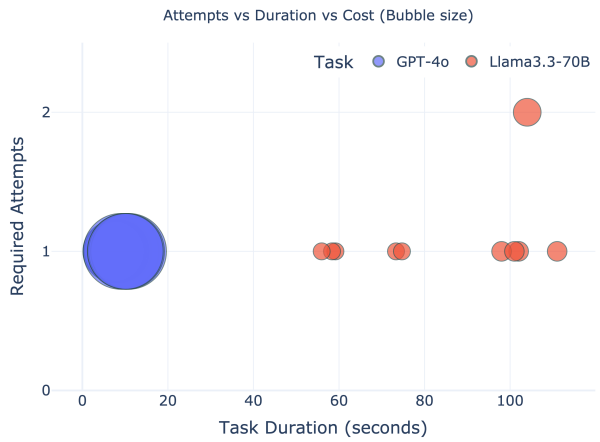

图 4. 匹配代理性能

确的匹配。图 4 显示了匹配 60 个需求所需的尝试次数和总时间。圆的直径代表成本。我们观察到 GPT-4o 在第一次尝试中成功匹配所有要求。Llama3.3 也实现了相同的目标，但有一些例外情况。此外，根据匹配任务的复杂性，GPT-4o 比 Llama3.3 快 6 到 10 倍。然而，GPT4-o 的成本大约是 Llama3.3 的 14 倍。

B. 代码生成代理性能

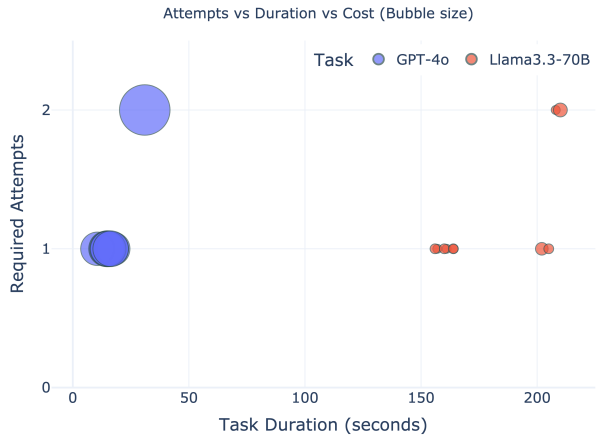

图 5. 代码生成代理性能

为了评估代码生成代理的性能，我们尝试根据匹配代理提供的 CFR 文档生成正确且可执行的代码。我们为 5 个 AP 和 5 个 gNB 生成代码，每个都使用不同的模拟供应商 API。图 5 显示了生成代码所需的尝试次数及总时间，而圆的直径则代表成本。我们观察到代码生成在第一次尝试中主要取得了成功。相比于匹配任务，编码任务更为复杂。在这里，Llama3.3 和 GPT-4o 之间的性能差距显著增加。在这种情况下，基于 GPT-4o 的代理比基于 Llama3.3 的代理快 10-14 倍，但成本也高 14 倍。

除了上述实验外，我们还尝试基于现有的 NF 功能生成增强的控制函数。此方法用于用户所需的控制功能与 NF 的能力没有完全匹配但有类似功能的情况。我们测试了两个控制函数：

- 信息年龄（AoI）感知的速率控制：此控制功能旨在在接入点上设置指定的数据速率，前提是给定的时间戳尚未过期。如果给定的时间戳超过接入点的系统时间，则不会应用速率设置。在这种情况下，接入点供应商支持速率控制，但缺乏信息年龄感知的速率控制。期望代码生成代理在供应商 API 之上添加一个封装器，该封装器可以将提供的时间戳与系统时间进行比较，并生成 API 服务器。
- 时间戳遥测收集：在这种情况下，控制功能要求所有返回的数据都带有时间戳。gNB 供应商支持数据收集但不包含时间戳。代码生成代理的任务是编写一个包装器围绕供应商 API，该包装器可以读取系统时间并将时间戳附加到返回的数据上。

我们发现，当需求中提供清晰且有结构的指令时，codegen 代理能够成功增强功能。平均而言，这项任务花费的时间是生成非增强控制函数的 3 倍，并且成本也是其 3 倍。然而，当我们尝试创建涉及生成多线程代码的更复杂的增强控制函数时，即使经过多次尝试，该代理也无法生成正确的代码。这只能通过手动干预来解决，具体包括修改原始提示。

C. 总体结果

总体而言，我们观察到 GPT4-o 提供了最佳的端到端性能。平均来说，生成一个包含 10 个控制功能的接口需要花费$0.04，并且具有 9.4 秒的端到端延迟。此外，对于给定代理的不同任务复杂度，GPT-4 显示出极小的延迟变化，而 Llama3 则表现出较大的延迟变化，即使是在类似的任务类型中也是如此。

VI. 限制

我们强调我们的工作的一些限制如下：

- 我们在合成数据上进行所有实验。这是由于大多数供应商并未公开他们的设备 API。尽管我们旨在尽可能现实地建模我们的合成数据（控制功能要求，供应商 API），但在实际场景中的性能可能会有很大

差异，因此需要在各种操作条件下进一步验证。代理的性能还取决于是否有结构良好且详细的特性和 API 文档 [10]。

- 考虑到该框架的实际部署，网络功能的安全性可能会受到特定于大型语言模型的威胁载体的影响。这些可能是直接或间接的提示注入或上下文窗口过载 [11]。在我们的实验中，我们观察到上下文窗口的大小会影响准确性。如果单个查询中的控制函数数量过大，模型开始产生不准确的输出 [12]。然而，这可以通过将其拆分为多个较小的查询来解决。
- 在匹配代理的情况下，没有地面实况供代理验证 LLM 模型的输出。因此，我们不得不使用另一个 LLM 来进行评估以验证输出。这可以扩展为使用一组专门的 LLM 来评估输出，但这也会增加成本。此外，由于 LLM 固有的不确定性，这种方法可能并非完全可靠。在实际部署中，生成的界面可能需要在一个沙盒环境中进行测试，然后再投入生产环境。
- 由于网络边缘的虚拟化，可以在边缘网络功能中部署大型 AI 模型。然而，这些模型对于 WLAN 接入点等设备来说可能太大。虽然可以通过 API 使用基于云的模型来缓解这个问题，但我们认为未来这些模型会变得足够小，可以部署在类似 AP 这样的设备上。

VII. 相关工作

NetconfEval [12] 引入了一个模型不可知的基准测试，用于评估 LLMs 在网络配置方面的性能。它评估了多个用例，包括开发路由算法、生成低级网络配置、将高级需求转换为正式规范和函数调用。

Netllmbench [13] 引入了一个用于在网络配置任务中对大语言模型进行基准测试的框架。它比较了各种大语言模型，包括 Llama3-70B，在操作速度和所需迭代次数方面的表现。

NAIL [14] 讨论了一种将用户高级指令转换为使用转译器生成的 P4 可执行代码的方法。它还能够持续监控网络以确保意图实现。

该系统 COSYNTH [15] 使用经过验证的提示编程来生成路由器配置。他们通过将 Cisco 路由器配置转换为等效的 Juniper 路由器配置来演示其应用。

在 [16] 中，作者讨论了多供应商 O-RAN 部署中的集成和兼容性挑战。他们提出了一种使用 WebAssembly 插件的框架，以实现互操作性和灵活的多供应商部署。

零接触服务管理（ZSM）[17] 框架可以集成 AI/ML 模型，自主检测并解决 6G 网络中的异常。

VIII. 结论与未来工作

我们提出了一种新型框架，用于在任意两组 NF 之间按需创建网络控制接口。我们通过利用 LLMs 开发一个多代理框架来实现这一目标。我们的框架为控制接口带来了灵活性和互操作性。我们相信在未来移动网络中，AI 代理可能会原生应用于网络的所有部分 [18]。使用 AI 按需生成控制接口是朝这个方向迈出的一步。

在这项工作中，我们仅关注控制接口，而不关注做出控制决策的应用程序（例如，xApps）。我们认为未来也可以使用人工智能生成决策应用程序。我们计划研究如何利用生成的控制应用和界面来为 6G 网络创建自主控制循环。

致谢

作者感谢德国联邦教育和研究部（BMBF）在 “Souverän. 数字化. 联网.” 计划中提供的经济支持，联合项目 6G-RIC（16KISK020K）。该项目还得到了德国联邦数字和交通部资助的 5G-COMPASS（19OI22017A）项目的支助。

参考文献

[1] “3gpp 23.501,” 2025, accessed: 2025-01-02. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.501/23501-j20.zip

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[3] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS ’20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[4] M. Wooldridge and N. R. Jennings, “Intelligent agents: theory and practice,” *The Knowledge Engineering Review*, vol. 10, no. 2, p. 115 – 152, 1995.

[5] M. Polese, L. Bonati, S. D’oro, S. Basagni, and T. Melodia, “Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges,” *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[6] "Ric platform - confluence/wiki," 2024, accessed: 2024-12-22. [Online]. Available: https://lf-o-ran-sc.atlassian.net/wiki/spaces/RICP/overview?homepageId=14123010

[7] "mosaic5g / flexric Â · gitlab," https://gitlab.eurecom.fr/mosaic5g/flexric, 2024, accessed: 2024-12-22.

[8] P. D. Thapa, A. Kappen, and J. Schulz-Zander, "Towards infrastructure-assisted wifi rate adaptation for converged networks with morpheus," ser. MobiArch '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3691555.3696828

[9] "Claude 3.5 sonnet," https://www.anthropic.com/news/claude-3-5-sonnet, 2024.

[10] S. Gunasekar, Y. Zhang, J. Aneja, C. C. T. Mendes, A. Del Giorno, S. Gopi, M. Javaheripi, P. Kauffmann, G. de Rosa, O. Saarikivi, and OTHERS, "Textbooks are all you need," 2023. [Online]. Available: http://arxiv.org/pdf/2306.11644

[11] Y. Liu, Y. Jia, R. Geng, J. Jia, and N. Z. Gong, "Formalizing and benchmarking prompt injection attacks and defenses," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 1831–1847.

[12] C. Wang, M. Scazzariello, A. Farshin, S. Ferlin, D. Kostić, and M. Chiesa, "Netconfeval: Can llms facilitate network configuration?" *Proc. ACM Netw.*, vol. 2, no. CoNEXT2, Jun. 2024. [Online]. Available: https://doi.org/10.1145/3656296

[13] K. Aykurt, A. Blenk, and W. Kellerer, "Netllmbench: A benchmark framework for large language models in network configuration tasks," in *2024 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2024, pp. 1–6.

[14] A. Angi, A. Sacco, F. Esposito, G. Marchetto, and A. Clemm, "Nail: A network management architecture for deploying intent into programmable switches," *IEEE Communications Magazine*, vol. 62, no. 6, pp. 28–34, 2024.

[15] R. Mondal, A. Tang, R. Beckett, T. Millstein, and G. Varghese, "What do llms need to synthesize correct router configurations?" in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, ser. HotNets '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 189 – 195. [Online]. Available: https://doi.org/10.1145/3626111.3628194

[16] R. Cannatà, H. Sun, D. M. Dumitriu, and H. Hassanieh, "Towards seamless 5g open-ran integration with webassembly," in *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, ser. HotNets '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 121 – 131. [Online]. Available: https://doi.org/10.1145/3696348.3696864

[17] M. Liyanage, Q.-V. Pham, K. Dev, S. Bhattacharya, P. K. R. Maddikunta, T. R. Gadekallu, and G. Yenduri, "A survey on zero touch network and service management (zsm) for 5g and beyond networks," *Journal of Network and Computer Applications*, vol. 203, p. 103362, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804522000297

[18] W. Tong, "A-ran, a-core, a-ue: The bridge to 6g," https://www.eucnc.eu/wp-content/uploads/2024/06/Keynote-Wen-Tong.pdf, 2025, accessed: 2025-01-09.