5.2 MAS Adapter

To better support agent discovery and selection, we propose a new component, the *MAS Adapter*. The MAS adapter is optional; simple agents (such as web services) may handle each enactment independently. Provided each role binding is used only within the context of an enactment, the information management provided by the protocol adapter is sufficient. The MAS adapter is responsible for remembering the contacts and the history of interactions with them beyond a single enactment.

The MAS adapter implements a registry of known agents. New entries are automatically added as agents are introduced. Each entry contains information about the protocols and roles that the agent is believed to support, based on the introduction. Each entry contains information about the roles that the agent has played and the corresponding enactment histories. Additional information can be added according to domain requirements, such as quality of service ratings. An agent can query its MAS adapter to find agents to enact protocols with.

6 EVALUATION

We now show how the above concepts can be applied toward capturing canonical role-binding patterns in practical decentralized applications.

6.1 Preconfigured Contacts

In peer-to-peer applications, it is common for the peers to be preconfigured with the knowledge of some bootstrap nodes to get the application going. Network applications are typically preconfigured with a DNS server for purposes of resolving domain names.

In our context, preconfigured contacts are those an agent is given before it is initiated. Preconfigured contacts are implemented in the agent internals and used to bind the initial ⌜out⌝ roles of the first protocols the agent enacts. This is a simple pattern, but it is a necessary component of all more complex patterns.

Listing 13: Preconfigured Contacts Example (JSON)

```
{"Seller":["http://storeA.com/agent","http://storeB.com/agent"],
 "Bank": ["http://bank.com/agent"]}
```

Listing 13 shows how contacts might be declared in a JSON file. Preconfigured contacts are not hard coded bindings; they are candidates that are bound to roles dynamically according to the protocol structure and agent logic.

6.2 Central Registry

The next level of complexity from a collection of preconfigured contacts is a central registry provided as a service by another agent. Discovering agents from a central registry is common in services. The Uber application, e.g., uses Uber's (the organization's) central registry of potential drivers to bind as the pickup driver in a particular transaction with a customer.

A central registry is simple in that it means that an agent configuration requires only a single registry connection. In addition, a registry enables more dynamic and scalable peer discovery than preconfiguration. As each agent comes online, they can enact a registration protocol, notifying the registry of their existence and willingness to perform specific roles. To enact a protocol, an agent can first query the registry to discover potential peers. As the example in Listing 14 shows, the pattern could be implemented as two protocols, where a single well-known agent uses its MAS adapter to remember and recommend contacts to other agents.

Listing 14: Central Registry

```
Registration(out A, R: role,
  out key ID, out endpoint, confirmation
  out set protocols: protocol) {
  out A -> out R: Register[out ID, out endpoint, out protocols]
  R -> A: Confirm[in ID, in endpoint, in protocols, out
       confirmation]
}

Discovery(out Q, R: role,
  out key ID, out protocol: protocol, out set agents: role) {
  out Q -> out R: Query[out ID, out protocol]
  R -> Q: Introduce[in ID, in protocol, out agents]
}
```

6.3 Peer Sharing

Peer sharing is characterized by the absence of any distinguished system nodes that support discovery. Peer-to-peer discovery and binding are common in MAS and distributed systems. For example, it is used in referral networks [34], in the Contract Net [32]), and in leader election protocols. Further, mesh networks and IoT-based systems typically invoke the peer-to-peer pattern.

To find a desired peer, each agent checks its own MAS adapter and queries its neighbors. The exact nature of the peer selection and query process is application-specific, but generally, each query will return more peers; either the desired peer will be among them, or they can be queried in turn.

Listing 15: Peer Discovery

```
Discover(out P1, P2: role, out key ID,
  out set protocols: protocol, out set neighbors: role) {
  out P1 -> out P2: Query[out ID, out protocols]
  P1 -> P2: Introduce[in ID, in protocol, out neighbors]
}
```

In the peer discovery protocol in Listing 15, there is no need for registration; each agent simply needs an initial bootstrap peer to connect to. Each time an agent asks a peer for neighbors, it naturally introduces itself and reveals the protocol(s) it is interested in.

7 DISCUSSION

We conceptualized the problem of instantiating a MAS in terms of binding the roles of the protocol that models the MAS. Although multiagent systems are typically thought of as being open in the sense of agents dynamically joining and leaving the system, MAS approaches have not paid sufficient attention to the operational aspects of the problem. Pippi demonstrates how MAS can be instantiated dynamically by showing how a protocol's roles are bound. Our approach supports dynamic (not hard-coded) and late (just-in-time) role bindings: not all roles in a protocol need to be bound before enacting the protocol. We discussed a possible realization of our approach and demonstrated how commonly used patterns can be captured in our approach.

Dastani et al. [14] consider compatibility between agent and role specifications as a basis for an agent's decision whether or not to play a role. Such internal decision making, although practically relevant, is outside the scope of Pippi. Pippi is concerned with public

aspects of decision making as reflected in protocol enactments. HAPN [33] supports dynamic role binding; e.g., an auction's winner is dynamically bound. It does not give a general metaprotocol-based approach that enables discovery and bindings of roles. Grenna et al. [19] extended JADE to enable agents to enact roles, but within the context of an organization rather than a protocol.

Günay et al. [20] propose a metaprotocol by which agents can reach agreement on the commitment protocol: All agents should accept that they will create the commitments involved. They assume role bindings (e.g., that there are customer and merchant agents). Thus, Pippi complements their work. McGinnis and Robertson [23] propose protocols as first-class abstractions for composing open MAS. Pippi agrees with their intuition and demonstrates how role bindings generated as information in the messages of one protocol (the metaprotocol) can be used as roles (the senders and receivers of messages) in another protocol via composition.

Multiagent systems are conceived of as open systems in the sense that agents can join and leave the MAS. Mazouzi et al. [22] give an early example of an agent wanting to join several groups and show that abstract specifications can be refined depending on requirements. Role binding, as we formalize here, is akin to "joining" a MAS. However, because parameter bindings and hence role bindings are immutable, "leaving" a MAS is not unbinding the role. Leaving would be captured by communicating that the agent has left the MAS. Immutability of information gives the immutability of events, which is necessary for realism. For example, once bound as a partner in some wedding, an agent is always a partner for that wedding. Divorce is not an unbinding of the agent to the partner role but a change in the normative relationships [9] between the partners, perhaps via the creation of a new MAS. For example, two divorced people may have joint custody of their children and may thus function together in a MAS, albeit not the same MAS as when they were married. Such ideas merit further study.

Minsky and Murata [25] discuss the robustness and manageability of a MAS modeled as a law-based society [24]. In considering robustness, Minsky and Murata bring up both static and dynamic role binding in the law, which is realized as a set of Prolog-like rules. The static bindings are given as facts, and the dynamic bindings are established via events referred to in the rules. Our approach is compatible with rule-based specifications (e.g., commitments or other norms [9]) but captures the general operational aspects of role binding in a decentralized system via protocols.

Ferrando et al. [17] consider the enactability [16] of protocols specified in a language for specifying execution traces, where the constraints specify message ordering. They evaluate the enactability of protocols under different infrastructure assumptions, e.g., with or without FIFO delivery. Ferrando et al. do not consider the problem of role binding. Other early work on formally specifying [29] and enacting [28] multiagent interactions also doesn't address role binding as an explicit concern.

Pippi's protocols can be enacted with the minimal set of infrastructure assumptions, namely that only sent messages be delivered. Further, Pippi's protocols can be interleaved without requiring to be composed (this is a property of information protocols [8]), which is not a possibility with trace-based approaches. Role binding for a protocol can be interleaved with the enactment of the protocol, as shown in the *Proposal* metaprotocol in Listing 3.

Chocron and Schorlemmer [6] consider the problem in open MAS where agents know they are participating in the same interaction (specified in temporal logic) but have different message vocabularies. They propose a method by which agents can dynamically align their vocabularies, that is, learn the mappings between their vocabularies. Pippi currently assumes a shared vocabulary (whatever is in the shared information protocols) but would benefit from alignment techniques in scenarios where the assumption does not hold. Arguably, aligning the "same" information protocol with different vocabularies is a more natural and challenging problem to address since information protocols can be more flexible than message ordering constraints specified in temporal logic.

Rocha and Brandão [27] apply multiagent systems to model dynamism in Internet of Things applications. They model devices via agents that enter and leave the system. Pippi enables realizing such dynamism in a general way with clean, modular representations.

JADE [2] supports the discovery of agents via a directory facilitator that provides a yellow pages service. JADE supports FIPA interaction protocols by providing endpoint implementations (classes and methods) that agents can use to implement interactions with others. However, JADE is not equipped to enable building multiagent systems off protocol specifications as Pippi's adapter and programming model enable. Further, discovery, although important, differs from role binding. In the wedding scenario, an agent can discover other agents looking for prospective partners via a directory service but would need to *bind* one of them to Proposee to enact Proposal. Briola et al. [4] demonstrate how JADE can be used to discover agents and implement a peer-to-peer system. As the foregoing examples demonstrate, Pippi supports both discovery and role binding and thus enables the realization of arbitrary applications as a peer-peer system.

Carriero and Gelernter's [5] tuple space approach for coordinating processes has been influential in multiagent systems. It features as the underlying coordination mechanism in CArtAgO [26], which itself is part of JaCaMo [3], which Baldoni et al. [1] use to support the implementation of commitment-based business processes. Tuple spaces (like logic programming) are attractive for their information-based abstractions (one works with tuples of information). However, they represent a shared-memory approach, which is not suitable for building decentralized MAS. It is widely argued that a tuple space decouples the readers and writers of information, but that argument holds only where the writers don't care who reads the information (as is the case in the classic "readers and writers" synchronization problem). In MAS, agents communicate with particular agents, e.g., the Proposer wants to communicate different information to the Proposee and to the Judge. Such coupling between agents is specified in the protocol. It would, however, be interesting to investigate if tuple spaces can be used to implement an agent's local state since it provides information-based abstractions.

ACKNOWLEDGMENTS

Thanks to the anonymous reviewers for their helpful comments. Christie and Chopra were supported by EPSRC grant EP/N027965/1 (Turtles). Singh was partially supported by the National Science Foundation under grant IIS-1908374.

REFERENCES

[1] Matteo Baldoni, Cristina Baroglio, Federico Capuzzimati, and Roberto Micalizio. 2019. Process Coordination with Business Artifacts and Multiagent Technologies. *Journal on Data Semantics* 8, 2 (June 2019), 99–112. https://doi.org/10.1007/s13740-019-00100-8

[2] Federico Bergenti, Giovanni Caire, Stefania Monica, and Agostino Poggi. 2020. The First Twenty Years of Agent-Based Software Development with JADE. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)* 34, 2 (2020), 36. https://doi.org/10.1007/s10458-020-09460-z

[3] Olivier Boissier, Rafael H. Bordini, Jomi Fred Hübner, Alessandro Ricci, and Andrea Santi. 2013. Multi-agent oriented programming with JaCaMo. *Science of Computer Programming* 78, 6 (June 2013), 747–761. https://doi.org/10.1016/j.scico.2011.10.004

[4] Daniela Briola, Daniela Micucci, and Leonardo Mariani. 2019. A Platform for P2P Agent-Based Collaborative Applications. *Software – Practice and Experience* 49, 3 (2019), 549–558. https://doi.org/10.1002/spe.2657

[5] Nicholas Carriero and David Gelernter. 1992. Coordination Languages and their Significance. *Communications of the ACM (CACM)* 35, 2 (Feb. 1992), 97–107. https://doi.org/10.1145/129630.376083

[6] Paula Daniela Chocron and Marco Schorlemmer. 2020. Vocabulary Alignment in Openly Specified Interactions. *Journal of Artificial Intelligence Research (JAIR)* 68 (May 2020), 69–107. https://doi.org/10.1613/jair.1.11497

[7] Amit K. Chopra, Samuel H. Christie V, and Munindar P. Singh. 2017. Splee: A Declarative Information-Based Language for Multiagent Interaction Protocols. In *Proceedings of the 16th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. IFAAMAS, São Paulo, 1054–1063. https://doi.org/10.5555/3091125.3091274

[8] Amit K. Chopra, Samuel H. Christie V, and Munindar P. Singh. 2020. An Evaluation of Communication Protocol Languages for Engineering Multiagent Systems. *Journal of Artificial Intelligence Research (JAIR)* 69 (Dec. 2020), 1351–1393. https://doi.org/10.1613/jair.1.12212

[9] Amit K. Chopra and Munindar P. Singh. 2016. From Social Machines to Social Protocols: Software Engineering Foundations for Sociotechnical Systems. In *Proceedings of the 25th International World Wide Web Conference*. ACM, Montréal, 903–914. https://doi.org/10.1145/2872427.2883018

[10] Samuel H. Christie V, Amit K. Chopra, and Munindar P. Singh. 2018. Compositional Correctness in Multiagent Interactions. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. IFAAMAS, Stockholm, 1159–1167. https://doi.org/10.5555/3237383.3237868

[11] Samuel H. Christie V, Amit K. Chopra, and Munindar P. Singh. 2021. Bungie: Improving Fault Tolerance via Extensible Application-Level Protocols. *IEEE Computer* 54, 5 (May 2021), 44–53. https://doi.org/10.1109/MC.2021.3052147

[12] Samuel H. Christie V, Amit K. Chopra, and Munindar P. Singh. 2021. Deserv: Decentralized Serverless Computing. In *Proceedings of the 19th IEEE International Conference on Web Services (ICWS)*. IEEE Computer Society, Virtual, 51–60. https://doi.org/10.1109/ICWS53863.2021.00020

[13] Samuel H. Christie V, Daria Smirnova, Amit K. Chopra, and Munindar P. Singh. 2020. Protocols Over Things: A Decentralized Programming Model for the Internet of Things. *IEEE Computer* 53, 12 (Dec. 2020), 60–68. https://doi.org/10.1109/MC.2020.3023887

[14] Mehdi Dastani, Virginia Dignum, and Frank Dignum. 2003. Role-assignment in open agent societies. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. ACM Press, Melbourne, 489–496. https://doi.org/10.1145/860575.860654

[15] Nirmit Desai, Ashok U. Mallya, Amit K. Chopra, and Munindar P. Singh. 2005. Interaction Protocols as Design Abstractions for Business Processes. *IEEE Transactions on Software Engineering* 31, 12 (Dec. 2005), 1015–1027. https://doi.org/10.1109/TSE.2005.140

[16] Nirmit Desai and Munindar P. Singh. 2008. On the Enactability of Business Protocols. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*. AAAI Press, Chicago, 1126–1131.

[17] Angelo Ferrando, Michael Winikoff, Stephen Cranefield, Frank Dignum, and Viviana Mascardi. 2019. On Enactability of Agent Interaction Protocols: Towards a Unified Approach. In *Proceedings of the 7th International Workshop on Engineering Multi-Agent Systems (EMAS) (Lecture Notes in Computer Science, Vol. 12058)*. Springer, Montréal, 43–64. https://doi.org/10.1007/978-3-030-51417-4_3

[18] Google. 2022. Protocol Buffers. https://developers.google.com/protocol-buffers/.

[19] Roberto Grenna, Matteo Baldoni, Guido Boella, Leendert van der Torre, Mauro Dorni, Andrea Mugnaini, and Valerio Genovese. 2008. Adding Organizations and Roles as Primitives to the JADE Framework. In *Programming Multi-Agent Systems (Dagstuhl Seminar Proceedings, 08361)*, Rafael Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah-Seghrouchni (Eds.). Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, Dagstuhl, Germany. http://drops.dagstuhl.de/opus/volltexte/2008/1639

[20] Akın Günay, Michael Winikoff, and Pınar Yolum. 2015. Dynamically Generated Commitment Protocols in Open Systems. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)* 29, 2 (March 2015), 192–229. https://doi.org/10.1007/s10458-014-9251-7

[21] Michael N. Huhns, Nigel Jacobs, Tomasz Ksiezyk, Wei-Min Shen, Munindar P. Singh, and Philip E. Cannata. 1992. Enterprise Information Modeling and Model Integration in Carnot. In *Enterprise Integration Modeling: Proceedings of the First International Conference*, Charles J. Petrie, Jr. (Ed.). MIT Press, Hilton Head, South Carolina, 290–299. https://doi.org/10.7551/mitpress/2768.003.0036

[22] Hamza Mazouzi, Amal El Fallah Seghrouchni, and Serge Haddad. 2002. Open Protocol Design for Complex Interactions in Multi-Agent Systems. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. ACM Press, Bologna, 517–526. https://doi.org/10.1145/544862.544866

[23] Jarred McGinnis and David Robertson. 2005. Dynamic and Distributed Interaction Protocols. In *Proceedings of the Workshop on Adaptive Agents and Multi-Agent Systems (Lecture Notes in Computer Science, 3394)*. Springer, Melbourne, 167–184. https://doi.org/10.1007/978-3-540-32274-0_11

[24] Naftaly H. Minsky. 1991. The Imposition of Protocols over Open Distributed Systems. *IEEE Transactions on Software Engineering* 17, 2 (1991), 183–195. https://doi.org/10.1109/32.67599

[25] Naftaly H. Minsky and Takahiro Murata. 2003. On Manageability and Robustness of Open Multi-agent Systems. In *Software Engineering for Multi-Agent Systems II, Research Issues and Practical Applications (SELMAS) (Lecture Notes in Computer Science, Vol. 2940)*. Springer, Portland, Oregon, 189–206. https://doi.org/10.1007/978-3-540-24625-1_11

[26] Alessandro Ricci, Michele Piunti, Mirko Viroli, and Andrea Omicini. 2009. Environment Programming in CArtAgO. In *Multi-Agent Programming, Languages, Tools and Applications*, Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni (Eds.). Springer, Dordrecht, Netherlands, Chapter 8, 259–288. https://doi.org/10.1007/978-0-387-89299-3_8

[27] Vladimir Rocha and Anarosa Alves Franco Brandão. 2019. A Scalable Multiagent Architecture for Monitoring IoT Devices. *Journal of Network and Computer Applications* 139 (Aug. 2019), 1–14. https://doi.org/10.1016/j.jnca.2019.04.017

[28] Munindar P. Singh. 1998. A Customizable Coordination Service for Autonomous Agents. In *Intelligent Agents IV: Proceedings of the 4th International Workshop on Agent Theories, Architectures, and Languages (ATAL-97) (Lecture Notes in Computer Science, 1365)*. Springer, Providence, Rhode Island, 93–106. https://doi.org/10.1007/BFb0026752

[29] Munindar P. Singh. 2003. Distributed Enactment of Multiagent Workflows: Temporal Logic for Web Service Composition. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. ACM Press, Melbourne, 907–914. https://doi.org/10.1145/860575.860721

[30] Munindar P. Singh. 2011. Information-Driven Interaction-Oriented Programming: BSPL, the Blindingly Simple Protocol Language. In *Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. IFAAMAS, Taipei, 491–498. https://doi.org/10.5555/2031678.2031687

[31] Munindar P. Singh. 2012. Semantics and Verification of Information-Based Protocols. In *Proceedings of the 11th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. IFAAMAS, Valencia, Spain, 1149–1156. https://doi.org/10.5555/2343776.2343861

[32] Reid G. Smith. 1980. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Trans. Comput.* C-29, 12 (1980), 1104–1113.

[33] Michael Winikoff, Nitin Yadav, and Lin Padgham. 2018. A New Hierarchical Agent Protocol Notation. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)* 32, 1 (Jan. 2018), 59–133.

[34] Pınar Yolum and Munindar P. Singh. 2005. Engineering self-organizing referral networks for trustworthy service selection. *IEEE Transactions on Systems, Man, and Cybernetics, Part A* 35, 3 (May 2005), 396–407. https://doi.org/10.1109/TSMCA.2005.846401