Agent-based Negotiation for Shared QoS Control in Distributed Cloud Services

Kaliappa Ravindran, Waldin Stone, and Arun Adiththan
Department of Computer Science
City University of New York (CUNY - City College), New York, NY 10031
Email: ravi@cs.ccny.cuny.edu; waldin@gmail.com; arunadiththan@gmail.com

***Abstract*—**Agent-level negotiations between an application and a cloud service provider (SP) need to consider two complexities: i) inaccuracy of computational models in capturing the behavior of a cloud-based system made up of diverse components and resources; and ii) disparity in the goals and priorities of an application and the SP in exercising a cloud-based system. The paper suggests the use of *trial actions* by an agent to learn about the system behavior and the opponent agent goals, as part of an effective negotiation strategy. The trial actions allow searching, alternations, and improvements on the system QoS space, towards a larger goal of optimal QoS control.

I. Introduction

The paper focuses on negotiations between the application and system agents to support quality of service (QoS) provisioning in a cloud-based distributed service support system. The agents face two knowledge uncertainties: i) QoS offering from service-provider (SP) with probabilistic guarantees due to resource fluctuations, and ii) generation of incomplete and imprecise QoS specs by an application. In a cloud setting, the problems are exacerbated due to the thirdy-party control of infrastructure and the diversity of application-level users. The agents get around problems (i)-(ii) by a persistent negotiation of the QoS needed and QoS offered.

QoS negotiation by software agents fits in a larger problem space of automated learning from past decision errors and adjusting the QoS needs and offerings therein to converge to a stable solution. We assume that an application is adaptive in its QoS expectations and the SP is responsive to support the changes in QoS needs with a suitable resource pooling. Figure 1 shows the system structure for agent-level negotiations to converge towards a solution: namely, system-level resource allocations needed for optimal QoS behavior. In the backdrop of hard-to-measure external factors and imprecise system models, agent-level negotiations are based on probabilistic assertions about the system behavior. A non-compliance to the negotiated level in a decision step is factored in a subsequent negotiation step between agents.

Software tools that employ game-theoretic, control-theoretic, and/or knowledge-theoretic techniques are available to support agent negotiations in complex environments. These negotiations are often based on a local and partial view of the state of an underlying distributed system [1]. Epistemic reasoning, for instance, is a knowledge-theoretic technique that determines the common knowledge available at the agents and allows a set of agent actions based on this knowledge. For

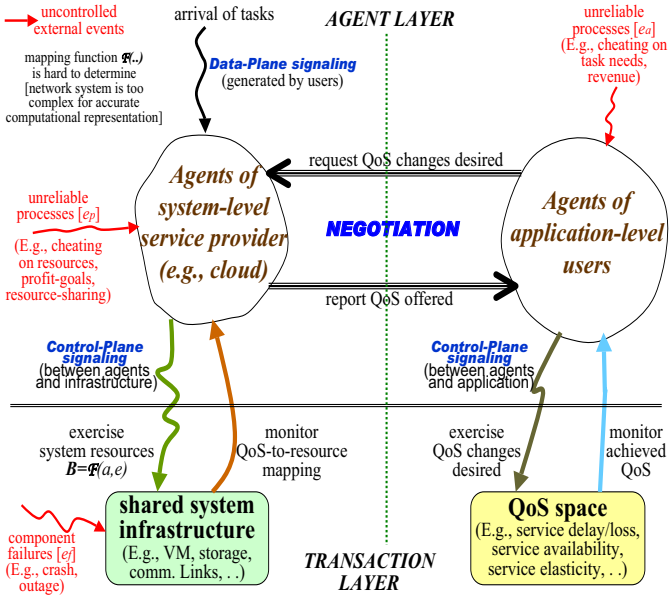

Fig. 1. System structure for QoS negotiations

example, knowledge of available bandwidth along a network path helps in improving the video transport QoS on this path. A control-theoretic approach that adjusts video send rate over the path in multiple send-and-observe cycles to bring the data loss below a threshold is also feasible. The control-theoretic approach provides a better QoS but is less resilient to the fluctuations in available bandwidth, when compared to the knowledge-theoretic approach.

Our negotiation approach employs *trial actions* on a network system under consideration. A trial action is proposed by an agent for orchestration on the network system to generate observable outcomes, which allows determining the course of subsequent negotiations. The generation of trial action(s) in a negotiation cycle is based on simplified approximate models of an intrinsically-complex network system. The model-estimated outcome of a trial action may however be different from the actual outcome observed, with the extent of deviation determined by how accurate is the model employed by the agent. The model-induced estimation error is factored in the generation of future trial actions, so that the error is progressively reduced in subsequent negotiation cycles. Each agent employs utility functions and policy rules in formulating a negotiation strategy

with other agents (say, self-centered or socially optimal) to reach what it considers as a global optimum.

Our concept of trial actions underscores two points. First, a trial action induces changes to the negotiation space based on the observed outcomes, which allows the participating agents to refine their strategies as the negotiation progresses. How the observed outcome of a trial action is incorporated by the agents to dynamically adjust their negotiation strategies depends on the adaptation and business logic programmed in them. Second, a trial action also enables the system progress towards a final solution. Even when there are no mid-point changes in the agent-level negotiation strategies, a sequence of trial actions leads to a solution. In this light, the notion of 'actions and trials' studied in [2] is more suitable for solution searches within a framework of state-transition based systems: because their notion treats the outcome of an action as simply a success or failure relative to an expected output. Whereas, our approach treats the deviation in actual outcome from an expected outcome as a continuum in the QoS space, which is amenable for probabilistic analysis. So, the level playing field for agent-based negotiations can be changed in a measurable way in various steps: thereby steering the system towards desired final outcome with a certain degree of assurance.

We explain trial action based agent negotiations in section II. Section III discusses service-layer primitives to encode agent actions. Section IV describes a study of agent-based latency control in a CDN. Section V summarizes our ideas.

II. Beating system complexity

The sequence of QoS changes occurring at the service interface as a result of trial actions is specific to the negotiation strategy. A self-centered negotiation approach, for instance, has both the SP and the application agents trying to maximize their local utilities in each negotiation step. Whereas, a social-welfare approach tries to optimize their combined utilities: thereby leading to a solution different from the self-centered approach [3], [4]. Regardless, our notion of trial actions enables the realization of both the approaches in a setting where the agents have a shared interest in using the service-support system, *albeit*, with different goals.

At the SP-end, mapping the QoS specs of an application onto a precise estimate of the resource needs in the underlying network infrastructure is intrinsically complex (e.g., third-party clouds with undependable resource guarantees). So, a promise of QoS offering from the SP is only as good as the accuracy of its resource estimates. This is further compounded by the inability of SP to precisely determine the available resources R^*_{av}. Nevertheless, the SP makes an approximate estimate of its available resources R_{av} to determine a feasible QoS offering during a negotiation.

The resource estimation R_{av} may be based on a combination of network probing and analysis with domain-specific software-implemented procedures. At the application-end, it is often difficult to determine the exact resource demands placed on the network system R^*_{dem}. Instead, an approximate estimate of the resource demands R_{dem} is feasible using simplified computational models of the network system. The agent-level negotiations are orchestrated with the resource estimates R_{dem} and R_{av}, in the face of estimation inaccuracies: $R_{av} \neq R^*_{av}$ and $R^*_{dem} \neq R_{dem}$. Thus, the problem of maximizing the agent utilities under the unknown constraint $R^*_{dem} < R^*_{av}$ has only a sub-optimal solution. The sub-optimality may manifest in the form of one agent gaining more at the expense of the other or both the agents gaining less than what is possible (say, due to under-utilization of network resources).

An agent can observe the actual system parameters in response to a pre-calibrated trial action during a negotiation round, and accordingly revise its resource estimates R_{dem} or R_{av}, as the case may be, for use in future rounds. Our premise is that trial actions allow an agent to refine its knowledge about the system model and learn about the opponent strategy. Since all the participating agents may resort to the try-and-learn method, the level playing field for agent-level negotiations gets elevated with improved knowledge.

III. Adaptation logic programmed in agents

We consider a service-layer that provides a *virtualization* of the infrastructure components in a network system S: say, a cloud. Algorithms in this layer map the component behaviors (including faults, outages, and errors) onto meaningful service-level behaviors. The current system state observable through the service interface provides a context for determining the resource control invocations from the agents.

An agent searches the QoS space with a sequence of incremental resource adjustments to reach a system-wide optimal behavior (in its view of the world). The resource adjustments are coordinated with the opponent agents via a series of negotiations. The multi-step search allows overcoming the complexity of QoS adaptation process (which arises due to combinatorial nature of infrastructure component interactions).

We envisage generic transactional operations such as 'roll-back', 'retry', and 'move forward' on the service interface state of S (see our earlier work [5]). These operations map onto the domain-specific algorithm-level activities in searching for an optimal operating point of S. The adaptation logic of an agent program, discussed in our present work, is conceivable at a meta-level even by non-specialists. It is then instantiated with domain-specifics to exercise S via stub procedures.

The agents may employ model-based engineering techniques to decide on the algorithmic actions meaningful in an observed state of S. One such technique is the Partially Observable Markov Decision Processes (PO-MDP): which is based state-machine analysis [6]. Epistemic reasoning: a knowledge-theoretic technique, also can determine the minimum common knowledge that is believed to be available at various agents [7]. Regardless of the method employed, the agent actions are often governed by the policies and rules of adaptation in the given application domain. The mapping from agent to system-level actions is often one-to-many.

For agents, the tracking error $\epsilon = (\gamma_q - \gamma'_q)$ is a meaningful generic metric that gets interpreted in a domain-specific way: where q is the QoS specs and q' is the QoS achieved. For

example, q' may be the observed query success rate in a replicated web service that processes client queries on a data repository to provide a timely response (say, $q' = 0.95$). Here, a quality improvement, i.e., a lower ϵ, can be achieved by increasing the number of VMs deployed for the service. In general, the quality metric ϵ at agent-level can be mapped onto one or more generic service-layer metrics.

In an earlier work [8], we have studied a model-predictive control-theoretic approach that allows agents to compute future trajectories and evolve action plans therefrom. In this light, our present work focuses on how agents can adjust their action planning based on internal strategies and the observed system response for a trial action. The strategy-driven adapting agents differ from fixed-role ones: e.g., network fault-detection [9].

IV. Agent-based QoS control in CDN

The usefulness of our transaction-style programming interface (TSPI) is exemplified with an illustrative scenario of agent-negotiated QoS control in a latency-adaptive CDN.

A. CDN QoS control functionality

Latency in delivery of a content to the clients, L, is a QoS parameter prescribed by clients through the CDN service interface. A service-layer algorithm places proxy nodes in the distribution tree set up over a cloud-based network infrastructure to store contents, whereupon a 'push' or 'pull' algorithm (denoted as SR and CL respectively) is employed to deliver content to the clients [10]. The QoS control is exercised with a suitable resource allocation (i.e., topological placement of proxy nodes) to keep the content delivery latency incurred L' below the latency specs L from the application. Figure 2 shows the layered view of a sample CDN.

The base topology created on the infrastructure shows the distribution tree, the nodes hosting proxy content p_a/p_b, and the nodes hosting clients and master server. When page p_a gets updated, it is pushed to the proxy nodes v, q; for p_b, the push occurs at nodes v, z — with the push to v occurring via node w. Clients C_1 and C_2 pull p_a, p_b from node v; whereas, client C_3 pulls p_a from q and p_b from node z. A crash of the link connecting v and y will trigger a tree reconfiguration to have C_1 pull p_a and p_b through node z instead of v. Likewise, if node v cannot provide content storage any longer, the role of node w may be upgraded as a distributor by leasing content storage capability in w.

Software agents at the application and CDN SP ends (A_agent and C_agent) execute the logic to iteratively search for a suitable resource allocation in the infrastructure that maximizes their interests. The agents invoke trial actions to dynamically change the resource allocation in search for an optimal solution that balances the latency L' against the resource cost. A trial action to improve the latency performance by $\Delta L'$ has the service-layer algorithm select a suitable placement of proxy nodes (from multiple candidate placements). The algorithm factors in the external environment parameters E^*: such as the dynamics of client workload and resource outages (e.g., crash of a proxy node), using CDN

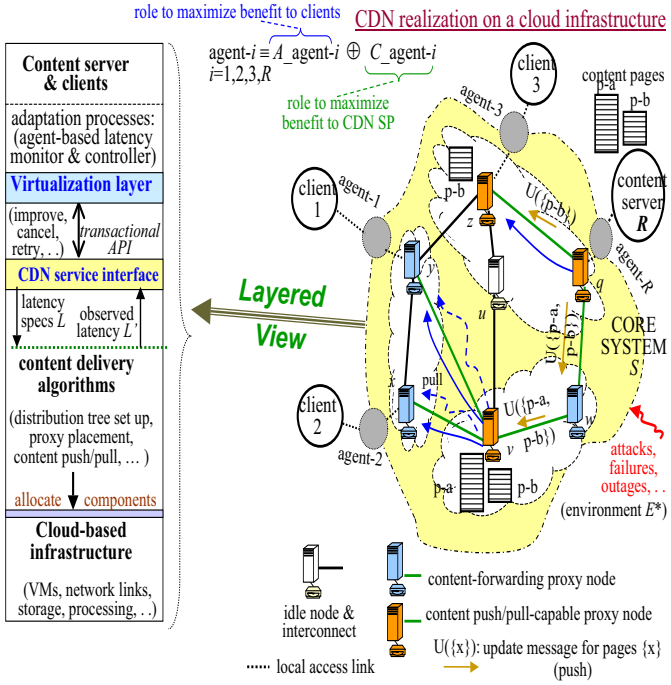

Fig. 2. Functional layers in a CDN

models. The optimization criteria programmed in A_agent and C_agent (i.e., utility functions), which may be different, are reflected in their negotiation strategies.

The generation of trial actions is tied to a model-theoretic assessment of CDN latency and overhead behvior $[L', O']$ in the future based on the currently observed system state at the service interface. Each agent compares its projected performance benefits relative to the current trajectory plan, and invokes a trial action if a better plan can be forecast. The benefit evaluation of a plan is based on the QoS utility functions programmed into the agents. The accuracy of a forecast however depends on how good is the computational model of CDN employed by the agents (see [8] for our M/M/G/1-type modeling of a CDN).

B. Negotiation-based QoS control

The tracking error ϵ depicts a reduction in the latency-based QoS relative to a prescribed level. This error is captured as:

$$\epsilon \equiv [1 - \frac{L}{k.L'}]|_{L'>L, k>1} \text{ for } \epsilon \in [0.0, 1.0].$$

For example, achieving a 80% guarantee that $L' < L$ when the desired guarantee is 95% indicates a tracking error $\epsilon = 0.15$. When $L' < L$, $\epsilon = 0$. The QoS-error tolerance parameter k is tied to the negotiation strategy internal to an agent: a higher k depicts a lower tolerance to the latency deviation $(L' - L)$. Here, a QoS improvement triggered by trial action (i.e., a lower ϵ) may be achieved by increasing the number of proxy nodes in the tree, increasing the content refresh rate (when the content changes), and allocating more network bandwidth. The A_agent goal is to lower its tariff for the CDN service while enjoying a higher QoS (i.e., propensity to lower ϵ).

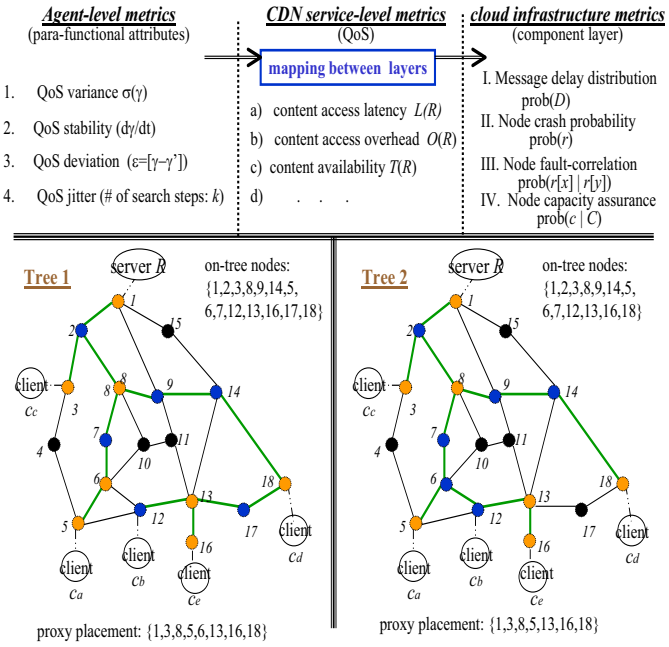

Fig. 3. Mapping of service metrics onto actions

Whereas, the C_agent goal is to improve the revenue accrued from clients while lowering the resource allocation cost (i.e., propensity to increase ϵ). Figure 3 shows sample observation metrics and their mapping onto system parameters.

C_agent and A_agent employ distinct utility functions: $U_c(L', L)$ and $U_a(L', L)$ respectively, to determine the client satisfaction in their own ways. The visible output of a trial action is $[L', O']$ experienced thus far. A function: $H'_{\text{pred}}(O', L')$, predicts the total overhead from the mid-point observations. The performance objectives are:

$$\Omega_c = l_1.f'_r(L_s, \mathrm{P}_{\mathrm{pred_m}}(L')) + l_2.[1 - U_c(\mathrm{W}_{\mathrm{pred_c}}(L'), L)] + H'_{\text{pred}}(O', L'); \quad (1)$$

$$\eta_a = l'_1.f'_r(L_s, \mathrm{W}_{\mathrm{pred_a}}(L')) + l'_2.U_a(\mathrm{W}_{\mathrm{pred_a}}(L'), L) \quad \text{subject to } \mathrm{W}_{\mathrm{pred_a}}(L') \leq L_C; \quad (2)$$

where Ω_c and η_a are the predicted per-client cost and satisfaction respectively (Ω_c is minimized and η_a is maximized). $\mathrm{W}_{\mathrm{pred_c}}(L')$ and $\mathrm{W}_{\mathrm{pred_a}}(L')$ depict a model-based prediction of latency from a mid-point observation L' at C_agent and A_agent respectively. A tariff reduction $f'_r(L_s, L') \in [0, 1.0]$ accrues when L' exceeds the specification of desired latency range $(0, L_s)$. The client dissatisfaction arising from a larger-than-L_s latency manifests as two sub-costs in Ω_c: lower tariff assessment and lower service reputation, with the constants l_1 and l_2 capturing their importance for C_agent relative to the overhead cost. Whereas, a client satisfaction arises from the actual L' experienced and tariff reduction, with their importance for A_agent captured by the constants l'_1 and l'_2. A_agent strives to keep L' within a delay-tolerance limit L_C when maximizing η_a, where $L_s \ll L_c$.

In each negotiation step, an agent maps the estimated latency and overheads onto a tree cost by determining the loss in utility of CDN system due to latency in excess of an acceptable threshold. We assume a trapezoidal shape of the utility function, which depicts a slowly diminishing usefulness of CDN service (and hence an increased cost) as the content-pull latency and overhead increase. The computed cost is then used in comparing the various trees and proxy placements generated during the search for an optimal placement.

The mapping of agent operations (such as retry and improve) onto the proxy-placements (executed in CDN service layer) is *one-to-many*. I.e., an agent-level operation can trigger one of many possible sequences of proxy placements, with a choice of actual trajectory based on the past search history. So, a mid-point change in the trajectory, as triggered by an agent action, requires the CDN service layer to maintain a state about the past search history: namely, the cost of various proxy-placements attempted. The one-to-many mapping, realizable with randomized techniques, improves the search process in its ability to find a near-optimal proxy placement. (instead of a statically built-in search mechanism).

V. Conclusions

The paper studied agent-level negotiations based on our notion of *trial actions* to manage the QoS of a complex network system. Trial actions allow an agent to dynamically learn about the system behavior and the opponent agent goals, as part of an effective negotiation strategy. The learning and negotiation, occurring hand-in-hand, purport to move the system towards an optimal behavior. Our trial action based negotiation mechanism is useful in cloud settings where agents tackle the problems arising from thirdy-party control of infrastructure and diversity of application-level users.

References

[1] Y. Moses. Knowledge as a Window into Distributed Coordination. In ICDCIT, LNCS 7154 (Ed. R. Ramanujam, S. Ramaswamy), Springer-Verlag, 2012.

[2] R. Niyogi and R. Ramanujam. An Epistemic Logic for Planning with Trials. In proc. Intl. Conf. on *Logic, Rationality, and Interaction*, Springer-Verlag, 2009.

[3] Y. Shoham and K. Leyton-Brown. Distributed Optimization. In Chap. 2, *Multi-agent Systems. Algorithmic, Game-Theoretic, and Logical Foundations*, Cambridge Univ. Press, 2009.

[4] R. Lin, S. Kraus, D. Tykhonov, K. Hindriks, C.M. Jonker. Supporting the Design of General Automated Negotiators. In *Innovations in Agent-Based Complex Automated Negotiations*, Springer-Verlag, vol.319, 2011.

[5] K. Ravindran. QoS Management by Competitive Agent-based Negotiation in Distributed Cloud Services. In proc. IEEE Intl. conf. on *Cloud Networking* (CLOUDNET), Luxembourg, Oct. 2014.

[6] A. Herzig, J. Lang, and P. Marquis. Action representation and partially observable planning using epistemic logic. In proc. *Intl. Joint Conf. on Artificial intelligence* (IJCAI), pp.1067-1072, M-K Publ., 2003.

[7] H.P. V. Ditmarsch, W. V. D. Hoek, and B.P. Kooi. Dynamic Epistemic Logic with Assignment. In proc. Intl Conf. on *Autonomous Agents and Multi-agent Systems* (AAMAS), Utrecht (Netherlands), 2005.

[8] K. Ravindran, Jun Wu, and A. Adiththan. Service Abstractions With Fault Virtualization for Distributed Network Infrastructures. In proc. IEEE/ACM Intl. Symp. on Distributed Simulation and Real-time Applications (DS-RT), Delft (Netherlands), Oct. 2013.

[9] M. Thottan and C. Ji. Fault Prediction at the Network Layer using Intelligent Agents. in proc. IEEE/IFIP Intl. Symp. on *Integrated Network Management*, pp.745-759, Boston (USA), 1999.

[10] Y. Chen, R. Katz, and J. Kubiatowicz. Dynamic Replica Placement for Scalable Content Delivery. In proc. *Intl. Workshop on Peer-to-Peer Systems*, LNCS-2429, Springer-Verlag, pp.306-318, 2002.