INVITED PAPER

Transitions: A Protocol-Independent View of the Future Internet

This article provides a taxonomy of an adaptation principle, named transition, which aims at increasing the flexibility and scale at which communication networks can be adapted.

By Bastian Alt, Markus Weckesser, Christian Becker, Matthias Hollick, Sounak Kar, Anja Klein, Robin Klose, Roland Kluge, Heinz Koeppl, Boris Koldehofe, Wasiur R. KhudaBukhsh, Manisha Luthra, Mahdi Mousavi, Max Mühlhäuser, Martin Pfannemüller, Amr Rizk, Andy Schürr, and Ralf Steinmetz

ABSTRACT | Countless novel approaches to communication protocols, overlay networks, and distributed middleware are published every year, yet the adoption of such novel findings in the global Internet landscape progresses at a slow pace. Many of such new communication mechanisms excel (only) under specific deployment conditions, while user mobility and application usage patterns lead to dynamic operation conditions. This mismatch is one reason that makes a wide deployment of new specialized mechanisms particularly hard as observed, for example, for multipath transport protocol extensions until the emergence of multipath transmission control protocol (TCP). This paper formalizes the concept of Transitions, i.e., a method to instrumentalize adaptivity at runtime in communication systems. It allows to exchange communication mechanisms in a running system to optimize the communication quality. In the following, we describe the building blocks required to: 1) capture the features and relations within a communication system and 2) express and optimize the decision making process in such a system. We show how this concept maps intuitively to the Internet model which makes a protocol-independent deployment of applications feasible in the future Internet.

KEYWORDS | Adaptive systems; communication networks; Internet; next-generation networking.

I. VISION OF TRANSITIONS IN A DYNAMIC COMMUNICATION ENVIRONMENT

For many decades, the design of communication systems has been closely tied to specific application scenarios. Building blocks of such systems such as communication protocols or overlay algorithms were particularly designed to fulfill a certain application objective. This led to a proliferation of what we denote mechanisms in the communication protocol landscape, i.e., there exits many protocols that essentially perform the same tasks, precisely, there exists many mechanisms that are functionally equivalent. We note that many of these specifically designed protocols are dominated by a few general-purpose ones. A prominent example of the subsequent ossification, i.e., the inability of changing these long-established general-purpose protocols, is the reliance on a few addressing and transport service schemes in today's communication systems.

Given the dynamics in future Internet communication scenarios such as autonomous driving and Internet of Things (IoT) use cases, we observe a need for inherent adaptivity, i.e., selecting the most appropriate combination of mechanisms subject to the context and environmental

Manuscript received June 30, 2018; revised November 26, 2018; accepted January 17, 2019. Date of publication February 25, 2019; date of current version March 25, 2019. This work was supported by German Research Foundation (DFG) as part of projects A1, A4, B3, B4, C1, C2, and C3 of the Collaborative Research Center 1053 Multi-Mechanisms Adaptation for the Future Internet. *(Bastian Alt and Markus Weckesser contributed equally to this work.)* *(Corresponding author: Bastian Alt.)*

B. Alt, M. Weckesser, S. Kar, A. Klein, R. Kluge, H. Koeppl, B. Koldehofe, W. R. KhudaBukhsh, M. Luthra, M. Mousavi, A. Rizk, A. Schürr, and **R. Steinmetz** are with the Department of Electrical Engineering and Information Technology, Technische Universität Darmstadt, 64289 Darmstadt, Germany (e-mail: bastian.alt@bcs.tu-darmstadt.de).

C. Becker and **M. Pfannemüller** are with the Chair of Information Systems II, University of Mannheim, 68131 Mannheim, Germany.

M. Hollick, R. Klose, and **M. Mühlhäuser** are with the Department of Computer Science, Technische Universität Darmstadt, 64289 Darmstadt, Germany.

Digital Object Identifier 10.1109/JPROC.2019.2895964

condition. This promises system responsiveness to scenario dynamics and uncertainty, which arises from unknown environmental and context conditions, such that applications do not perceive a change of communication quality. To this end, we require a concept to describe such an adaptive system and to express and optimize the mechanism decision making process under uncertainty.

In this paper, we build on a previous body of work that generally describes the concept of transitions, i.e., "the functional replacement of a mechanism by a functionally similar or equivalent other mechanism in a running communication system without causing an error" as defined in [1]. We also use the definition of the term mechanism from [1] which states that "a mechanism is a confined conceptual element of a networked system that is bound to a realization as cooperating functional units." Examples of mechanisms include services in the open systems interconnection sense or simply communication protocol functions (see [1, App. B] for more details). Here, we provide a tutorial-style approach to a framework that describes and optimizes reconfigurable communication systems at runtime. First, we describe how to capture the relevant features of a communication system including their relationships and restrictions before, second, showing how to optimally decide on reconfigurations in such a system under uncertainty. In the following, we describe a dynamic communication scenario to motivate the need for mechanism transitions.

A. Requirement Driven Adaptation in Dynamic Environments

To show the requirements for adaptation in a highly dynamic environment, we consider next a heterogeneous wireless network as discussed in [2] consisting of mobile nodes that possess multiple communication interfaces. The nodes form a distributed system as typically found in IoT, connected industry automation scenarios and autonomous driving. For example, in the autonomous driving scenario, these nodes collectively perform tasks of different complexity subjects to various requirements that range from basic tasks such as congestion detection to compound tasks such as collaborative maneuvers. The collaborative tasks pose different requirements on the communication quality, e.g., in terms of quality of service, reliability, or simply the used protocol. Hence, as the nodes possess multiple communication interfaces that map, e.g., to physical layer technologies such as long term evolution (LTE), WiFi, and mmWave, each node may decide on the utilized protocols on top of a subset of the available communication interfaces.

Next, consider the task of detecting a vehicle traffic congestion in an autonomous driving scenario. Each node, here, a vehicle, may process an operator, i.e., a function, that leads to a detection of an event of interest. An example of such an operator can be a filter that detects slow movement in correlation with a high density of vehicles on a road segment. In essence, the nodes collaborate to process multiple operators subject to the performance requirements of the collaborative task, e.g., reliability requirements for cooperative awareness tasks as described in [3]. Note that the underlying communication mechanisms highly influence the ability of the nodes to fulfill these performance requirements. Hence, in any of the mentioned scenarios, the nodes exchange information for distributed processing and decision making, under the constraint of meeting the performance requirements. Since these nodes are exposed to a dynamic environment they need to adapt the deployed communication mechanisms to meet the aforementioned requirements. Next, we will outline how these nodes can make use of the concept of transitions to adapt to the dynamic environment of the described scenario.

B. Transition Decisions

In the following, we consider individual transition decisions within the dynamic communication scenario described in Section I-A. More specifically, we consider two types of transition decisions, i.e., decisions that affect individual nodes, as well as transition decisions that collectively impact multiple nodes. The first type of transition decisions includes the decision to map data to the available wireless interfaces while the second type includes, e.g., the collective decision of all nodes to utilize a topology control (TC) algorithm to control the mobile communication network.

First, the mapping of data packets to available communication technologies, which we denote as scheduling, can be described using different mechanisms that include proportional allocation, full data replication, preference-aware data allocation and gradient descent techniques to name a few. Second, the collective node decision to utilize a TC mechanism that is suitable for the used wireless communication mechanism entails that, e.g., cone-based TC mechanisms are used for directed mmWave links in contrast to triangle-based TC mechanisms for WiFi links. Finally, note that due to the distributed nature of collective task processing in such a network an operator placement (OP) problem arises, i.e., the question of the assignment of an operator to a node. In the aforementioned communication scenario, some OP mechanisms may be suitable for distinct wireless technologies. For instance, centralized OP mechanisms for LTE communication while decentralized OP mechanisms for *ad hoc* networks.

Now, consider a dynamic communication environment where the nodes experience, for example, a degrading quality of the mmWave links and decide to deploy a transition from mmWave to a combination of WiFi and LTE. Here, the nodes need to use a scheduling mechanism that transmits packets on the two newly available communication mechanisms. As a consequence of changing the communication mechanism, a switch from a decentralized OP mechanism to a centralized one is required. Finally, the nodes switch from the cone-based TC mechanism to a triangle-based one.

To be able to decide on the transitions that should be executed in such a dynamic communication environment, we observe that a logic is required for reasoning on transitions and consequently optimizing over them. As outlined in [1], this transition control logic follows the monitoring analysis planning execution (MAPE) principle with the phases of monitoring, analyzing, planning, and executing [4]. Monitoring in this class of scenarios is measured covariates that capture the environment dynamics such as latencies, round-trip times, and packet losses. In this context, the analysis phase consists of nodes comparing measurement results to the requirements of the execution, e.g., maximum latencies or minimum throughput. The planning phase comprises planning a switching procedure, i.e., a mechanism transition, if it is anticipated that the requirements will not be met. This transition aims to reach a system state which is within the defined requirements. Such a procedure is, for example, the exchange of a single or multiple mechanisms, e.g., the physical interface used or the scheduling algorithm. The execution phase must, e.g., make sure to execute the plan on the correct nodes, if the plan only affects a part of the distributed system.

Building on the scenario above of a transition-enabled communication system, we discuss in the next section, a general framework to describe the communication system, the dynamic environment and an optimal adaptation behavior in the view of the concept of transitions.

C. Paper Structure

This paper is structured as follows. After providing a vision of transitions in a dynamic communication environment in this section, we introduce the key building blocks that contribute to a formal description of a transition-based communication system in Section II. In Section III, we provide a mapping of the explored framework to the well-known five-layer Internet model. We discuss the capabilities and caveats of the proposed modeling framework in Section IV before reviewing related concepts on modeling reconfigurable systems in Section V and concluding this paper in Section VI.

II. BUILDING BLOCKS OF A TRANSITION MODEL

In this section, we describe an approach to modeling a transition-based communication system. We first provide a high-level overview before discussing the details of its key building blocks. We note that it is crucial to find an appropriate level of granularity and abstraction to formally describe such a system. A model that is too fine-granular and tries to accommodate all aspects of an implementation may well suffer from analytic intractability. Therefore, it is important to first identify the indispensable features of the system and then describe the overall system behavior in terms of those features.

To model communication system properties, constraints, and capabilities, we use the well established and widely used methodology of dynamic software product line (DSPL) engineering [5]–[7]. Having identified the key features of the system, we implement the concept of transitions in the mathematical framework of Markov decision processes (MDPs) [8]. To this end, an optimal system behavior is found by the means of utility design. Next, we describe the modeling approach, a detailed discussion on several aspects of capabilities and scalability of the model is further discussed in Section IV.

A. Dynamic Software Product Lines

DSPLs constitute a methodology to capture a large configuration space concisely and to specify runtime variability of adaptive systems [6], [7]. Existing DSPL approaches provide means to specify adaptability in terms of valid configurations with reconfiguration decisions triggered at runtime. Feature models are used to specify the configuration space of the adaptive system by domain experts during a well-structured domain analysis process. A feature refers to a domain abstraction from the problem space of the DSPL, corresponding to a binary configuration option of the adaptive system, which is either selected or deselected. In the context of a transition-based communication system, a feature may refer to mechanisms, components of mechanisms, or their parameters. For example, in the communication scenario discussed in Section I, the mechanism "mmWave" is represented as a feature that might be active (selected) or inactive (deselected). A configuration is a set of selected features and corresponds to a valid communication system state. In our communication scenario, the configuration containing the selected features Wireless, LTE, Scheduling, Replication, and Context constitute a valid communication system state. The features representing configuration options of the communication system will be outlined over the course of this paper.

Feature diagrams as part of the feature-oriented domain analysis [9] restrict the configuration space by capturing dependencies between features. These features are organized in a tree structure to represent a decomposition relationship between a parent feature and its children. A decomposition relationship between a parent feature and its children has one of the four types: mandatory, optional, or, and alternative. Furthermore, cross-tree constraints describe functional relationships between features outside the tree structure and are either of type require or exclude or might be given as propositional formula over features. In our example, we require that whenever feature GeographicallyDistributed is selected, feature Decentralized is present in the configuration as well.

Feature attributes capture measurable nonfunctional characteristics of features [10]. The domain of an attribute represents the space of possible values of the attribute [11]. A domain is either discrete or continuous and is characterized by an interval that imposes lower and upper bounds on the attribute values (e.g., $k \in \mathbb{R} \cap [1, 2)$). Note that relationships between attribute values and features constrain the configuration space. We consider

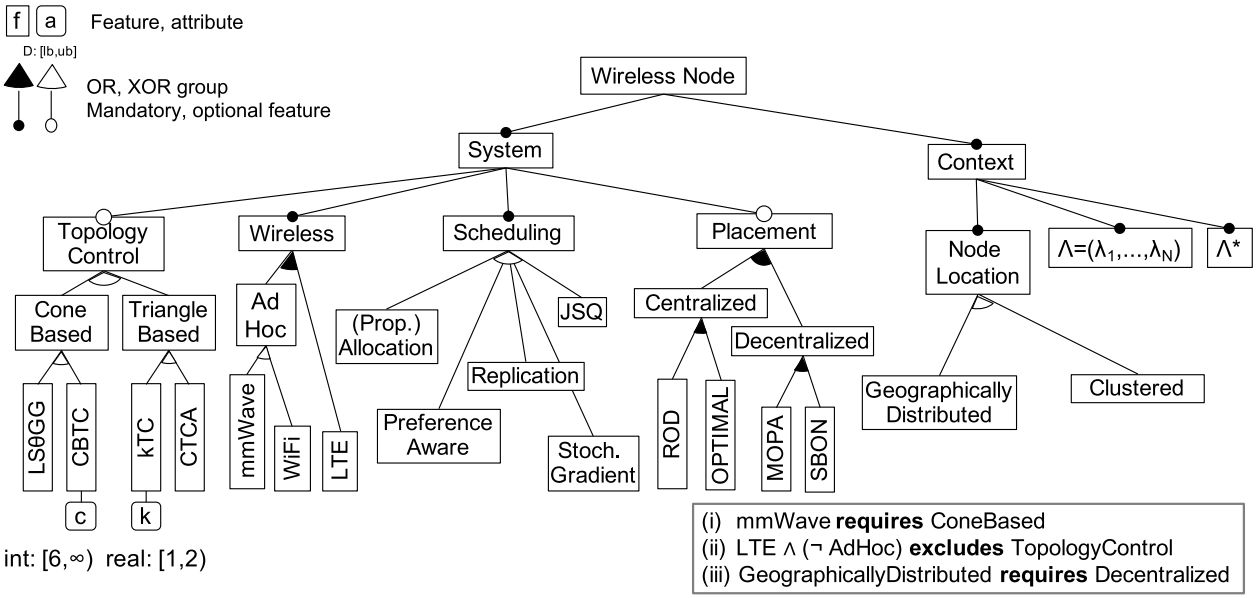

Fig. 1. ***Feature diagram of the communication scenario in Section I. The tree structure shows the dependencies of the system and context features, by using logical operators. Here, three exemplary cross-tree constraints are given as a list. The system configuration is described by the system subtree; the communication system state is described by the entire context feature model.***

constraints of the form

$$e_{lhs}\ [\text{requires} \mid \text{excludes}]\ e_{rhs}$$

where the left-hand side and right-hand side expressions, e_{lhs} and e_{rhs}, are the propositional expressions that are recursively defined as $\neg e \mid e_1\ [\wedge \mid \vee]\ e_2 \mid [f \mid a]\ [< \mid > \mid =]\ v$ with the (binary) feature f, the attribute a, and the value v, which is within the domain of f or a, respectively.

Fig. 1 shows the feature diagram that specifies valid configurations of the nodes considered in the example provided in Section I. The OR-group below the feature Wireless specifies that a device communicates using one or more physical network interfaces at a time. In the motivating example, we consider infrastructure-based communication (LTE) and *ad hoc* communication (AdHoc) using omnidirectional (WiFi) and directional antennas (mmWave). Omnidirectional antennas simplify multicast or broadcast communications at the cost of interference and range, whereas directional antennas provide higher gain and less interference, but only in a specific direction. We note that infrastructure communication is robust if the network is highly dynamic, e.g., nodes under high, unstructured mobility (e.g., within unstable neighborhoods). *Ad hoc* (device-to-device) communication may be more suitable under low or structured mobility (e.g., swarms of nodes) if low latency communication is required, e.g., if the infrastructure-based communication suffers from congestion.

1) Communication Interfaces and Topology Control: Considering Fig. 1, we observe that, if AdHoc is active, topology control aims at reducing the length of the longest incident link in the logical topology spanned by the communicating nodes to allow for a reduction of the transmission power or for using a more efficient and less robust modulation scheme [12]. If mmWave is active, topology control helps to limit the computational complexity during the expensive pairwise negotiation of channel parameters [13]. If only LTE is active, topology control should be inactive. This restriction is specified by cross-tree constraint (ii) in Fig. 1.

We further distinguish two large families of topology control mechanisms [14]: triangle-based (TriangleBased) and cone-based topology control mechanisms (ConeBased), from which a specific topology control mechanism is exclusively selected. In general, a triangle-based topology control mechanism removes a link from the logical topology if this link is the weight-maximal link of a triangle in the input topology and if the triangle fulfills some additional algorithm-specific predicate (kTC [15] with parameter k and CTCA [16]). A cone-based topology control mechanism partitions the area around the node into disjoint cones and only preserves a fixed number of links per cone based on an algorithm-specific predicate (CBTC [17] with parameter c and LSΘGG [18]). Only cone-based topology control mechanisms are suitable in conjunction with directional communication as found in mmWave, as specified by cross-tree constraint (i) in Fig. 1.

2) Collective Task Execution: In light of the scenario described in Section I, multiple nodes may be required to fulfill a collective task that is described by a set of operators, i.e., functions, that are placed onto the different nodes. Hence, Placement is represented as an optional system feature that becomes available in the view of distributed processing, depending on the scenario. The OR-group below Placement specifies that one or more placement mechanism can be active in parallel. Different users may

be interested in distinct performance objectives that results in selecting more than one placement mechanisms at a time. In addition, the selection of Placement can be task dependent (e.g., traffic congestion and shortest route). Considering the feature Placement for a traffic congestion detection task for in-network distributed processing as described earlier in Section I, the feature Placement is divided into two categories depending on the location of the processing nodes: Centralized and Decentralized placement. In general, centralized placement mechanisms assume a global knowledge of the network at a central placement coordinator, which then determines the assignment of a task. In contrast, decentralized placement does not assume such global knowledge; instead, it uses local information shared between the neighbors to determine the task assignment.

Now, if the feature AdHoc is active, placement aims at nodes that are within the same administrative domain to allow for efficient device-to-device communication. In this case, considering clustered nodes at close distance will be suitable, which leads to activating the feature of a centralized placement coordinator to deploy corresponding placement mechanisms such as ROD [19] or OPTIMAL [20].

3) Context Feature Models: Context variability plays a key role for DSPLs because system configurations must adapt to changes in their context [5]. Context feature models (CFMs) allow to model relevant properties of the context as context features and to specify their dependencies to system features using cross-tree constraints [21]. In Fig. 1, system features and context features are organized in separate branches of the feature model. Features of the context branch are selected to reflect a given context. In our scenario, the context feature NodeLocation reflects the geographical distribution of the communicating nodes. The feature $\Lambda = (\lambda_1, \ldots, \lambda_N)$ captures the parameters describing the quality of the communication mechanisms. The context configuration is the partial configuration that represents all selected features of the context branch. Accordingly, the system configuration is the partial configuration that represents the system branch. When the context changes, its configuration is adjusted accordingly, and the system configuration may need to be adapted.

B. Markov Decision Processes

Given a dynamically changing communication environment, we require transition plans that define transitions between the available communication mechanisms. A transition plan defines reconfiguration decisions based on the communication quality and anticipated context changes. In general, for a given context, a large number of valid system configurations exists which are, however, of various communication qualities. In general, one aims at transition plans that are optimal in the sense that they maximize a desired objective in such dynamic environments. Such an objective is the choice of the communication system designer and can include diverse goals, such as minimal energy cost for a mobile device or maximum throughput in scheduling decisions.

The framework of MDPs lends itself naturally to capture the adaptation of system configurations, as specified in Section II-A. MDPs are an established method for optimal decision making in uncertain environments (see [8]). Formally, an MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, P, \Phi, \gamma \rangle$.

The state space $\mathcal{S} = \mathcal{M} \times \mathcal{C}$ contains all relevant information of the communication system with the set of system configurations $\mathcal{M}$ and the set of contexts $\mathcal{C}$. A state $s = (m, c) \in \mathcal{M} \times \mathcal{C}$ of the communication system contains, therefore, the system configuration m and the context configuration c of the DSPL model of Section II-A. For example, in the communication scenario of Section I, one state of the MDP is the system configuration m with a selected LTE and WiFi feature with the Scheduling feature "join the shortest queue" (JSQ). Here, the context configuration c includes the context feature Λ, which parameterizes the quality of the two active wireless technologies.

The set of actions $\mathcal{A}$ contains the finite set of possible adaptation actions. This means that an action $a \in \mathcal{A}$ contains the binary information of selecting and deselecting different feature combinations. Therefore, the context configuration and system configuration can be changed. For example, the selection of the LTE feature and the selection of the JSQ scheduling feature can be seen as an action that leads to a transition of the communication system.

The transition probabilities $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ describe the stochastic behavior of the whole communication system. For a given action a and a state s, the probability $P(s' \mid s, a)$ is a probability distribution over s' and describes the probability to move from state s to state s' with action a. Exploiting the Markov property and assuming that the context is not influencing the adaption of the system configuration, the probability of switching a state can also be expressed as $P(s' \mid s, a) = P(m', c' \mid m, c, a) = P(m' \mid m, a)P(c' \mid m, c, a)$. We assume that the switching between system configurations is carried out deterministically. Hence, $P(m' \mid m, a) = 1$ if the system configuration m is changed under action a to a new valid system configuration m', as described in Section II-A. Therefore, the transition probabilities encode valid system configurations by setting transition probabilities of certain invalid feature combinations to zero. Note that we assume that the underlying Markov chain is ergodic. Consider, for example, the scenario depicted in Fig. 1, where an invalid configuration in which both features mmWave and TriangleBased TC are selected. Since this partial configuration is forbidden by the feature model, every corresponding transition probability to this configuration is set to zero. In addition, we assume that the context changes probabilistically according to a Markov chain. This means that, for the MDP, the transition probabilities are assumed to be Markovian $P(s_{t+1} \mid a_t, s_t, \ldots, a_0, s_0) = P(s_{t+1} \mid a_t, s_t)$. We further assume time-homogeneity $P(s_{t+1} = s' \mid a_t = a, s_t = s) = P(s_t = s' \mid a_{t-1} = a, s_{t-1} = s)$. This stochastic context change naturally captures dynamic

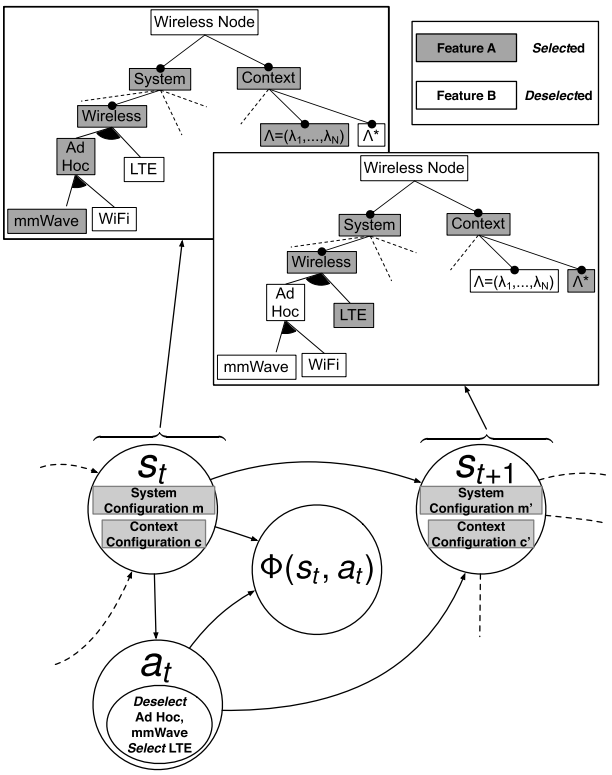

Fig. 2. *Probabilistic graphical model for the adaptation behavior of the communication scenario from Section I. The system configuration m (mmWave interface selected) is transitioned to system configuration m' (LTE interface selected) which leads to the stochastic context change of context configuration c (context feature ˜ selected) to context configuration c' (context feature ˜* selected). The boxes show a snippet of the corresponding CFMs of the states of the communication system.*

communication environments, where the stochastic context change is quantified by $P(c' \mid m, c, a)$. Note that the context features change according to a Markov Chain. This is a reasonable modeling assumption given that, e.g., Internet traffic can be modeled using the Markov modulation or mixture modeling [22]. For context changes that do not possess the Markov property, the context configuration can be extended to contain the sufficient statistics of the history, such that the process conditioned on the history becomes Markovian. Since this can lead to large state spaces, approximation techniques to solve the MDP may be used in such cases. In Fig. 2, we show the time evolution of an example transition from one valid system configuration to a new configuration in form of a probabilistic graphical model. The corresponding state space of the underlying ergodic Markov chain is shown abstractly in Fig. 3.

The performance function $\Phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ models the performance of transitions, i.e., taking actions in given states. Note that the performance can be positive or negative where the latter is understood as a type of cost. We introduce a discount factor γ which is a scalar variable that models the tradeoff between short- and long-term performance. The design of the performance function leads to varying adaptation behavior of the communication system. For example, a performance function can incorporate the cost of an adaptation decision balanced with the performance gain $\Phi(s, a) = \Phi(m, c, a) = w_1\Phi_g(m, c, a) - w_2\Phi_c(m, a)$. Here, the weight constants w_1, w_2 balance the performance gain Φ_g of adapting the system configuration m under context c by taking action a, and the performance cost Φ_c of adapting the system configuration m by taking action a. An illustrative example is balancing the performance gain Φ_g of a higher throughput when transitioning to a different better suited wireless interface and the adaptation cost Φ_c in terms of additionally required energy to perform the transition. A more detailed discussion of this design space is given in Section II-C. For now, we assume the performance function to be given by domain knowledge of the system designer.

Next, we describe the dynamics of the MDP by the following generative process. The communication system starts in an initial state $s_0 \in \mathcal{S}$. At every time step t, the communication system chooses an action a_t which leads to a new state s_{t+1} with probability $P(s_{t+1} \mid s_t, a_t)$ where the communication system achieves the performance $\phi_t = \Phi(s_t, a_t)$. We define a deterministic transition plan/policy as a mapping $\pi : \mathcal{S} \to \mathcal{A}$. This defines, in principle, a set of possible adaptation rules. For example, consider the state where the mmWave feature is selected and the context implies a low quality of this communication channel. In this state, a transition plan can be the action to deselect the mmWave feature and select the LTE feature, which, in turn, leads to a new valid system configuration given the cross-tree constraints shown in Section II-A. Using an infinite horizon MDP, the optimal

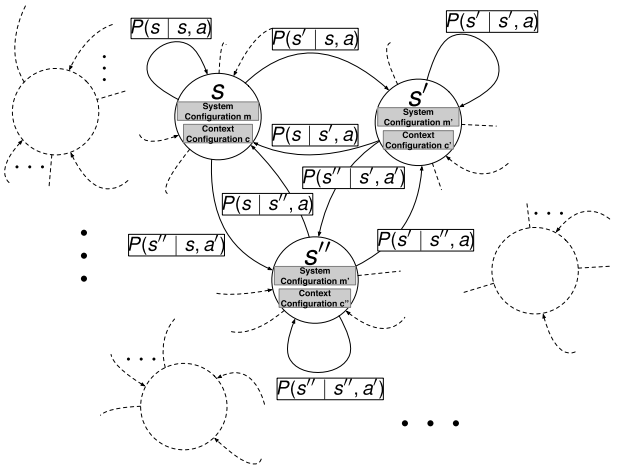

Fig. 3. *Markov chain state space view of the configuration adaptation behavior. Here, a snippet of the state space is shown. Action a and a' denote adaptations to reach system configuration m and system configuration m', respectively. The context change is captured as the result of the state transitions according to the transition probabilities.*

transition plan incorporates the prediction of future performance which makes the adaptation rules proactive in the sense of predicting the stochastic time evolution of the communication system. The optimal transition plan π^* is the transition plan that maximizes the expected discounted performance as

$$\pi^*(s) = \arg\max_{\pi} \mathrm{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \Phi(s_t, a_t) \mid s_0 = s\right], \quad \forall s \in \mathcal{S}.$$

This is possible since, under the assumption of finite state and action spaces, the optimal transition plan can be expressed as a deterministic mapping. The value of the optimal transition plan is found by solving the Bellman equation

$$V^*(s) = \max_{a}\left\{\Phi(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^*(s')\right\}.$$

The optimal transition plan $\pi^*(s)$ of a state s is then the action that maximizes the value $V^*(s)$ at that state. In the light of a transition-based communication system, the optimal transition plan is the set of adaptation rules for the communication system that achieves the highest overall quality by optimizing over the communication system utility. The calculation of an optimal transition plan can be achieved by numerous algorithms, such as, e.g., value Iteration [23], policy Iteration [24], linear programming [25], or probabilistic inference [26].

Since we assume the transition probabilities to be known, the solution to the MDP is referred to as planning. Note that the transition probabilities can also be inferred. Taking the scheduling decision as an example, path conditions may be described, more formally, as a random vector denoting the path-specific packet transmission times, e.g., using LTE, WiFi, and mmWave. Optimal scheduling decisions, given known individual path conditions or estimates, thereof are shown, e.g., in [27] and [28], respectively. Frameworks that enable the execution of such optimal scheduling decisions by changing packet-to-path mappings at runtime are given, e.g., in [29].

In the case of uncertain transition probabilities, Bayesian learning provides an approach for optimal planning [30]. When offline inference of the transition probabilities is not available, another promising approach to solve the MDP is reinforcement learning [31]. However, a challenge may arise if the context is not directly observed but, instead, only noisy monitoring observations are available. In this case, solution techniques of partially observable MDPs provide a viable approach. Since only small scale problems can be solved exactly, usually approximate solvers are used. Another aspect to note is that, in this paper, we only discuss finite-state Markov processes. However, for the incorporation of continuous-valued states, a multitude of solvers have been discussed. Such solution methods usually incorporate some function approximation of the value function and/ or the policy. We further discuss several aspects of scalability in Section IV.

C. Utility Design and Decision Making

As introduced in the previous section, utility functions are necessary for quantifying the performance of individual configurations of the transition-based communication system as well as deciding on which mechanisms to use. Utility functions trade the benefit of using a mechanism for its cost while the maximization of the utility function simply determines which mechanism (from a set of possible mechanisms) has to be used. When multiple mechanisms have equivalent benefit, the utility maximization problem is turned to a cost minimization problem [32], [33]. For example, considering the same data rate for LTE and WiFi, a cost minimization problem would be choosing the configuration that results in lower energy consumption.

In the light of the communication scenario described in Section I, we consider collective task executions in the described distributed communication system. Here, the goal of the nodes to maximize their own performance rationally and selfishly can be expressed using a game-theoretic point of view. In accordance with the definition of the set of actions in Section II-B, the action of a node i in the game, denoted by a_i, is to choose a mechanism among all possible actions (mechanisms) in the action set of the node, denoted by $\mathcal{A}_i$. We denote the set of nodes, i.e., players of the game, by $\mathcal{N}$. To this end, the joint action set of the game that represents the solution space of the game is given by $\boldsymbol{\mathcal{A}} = \sqcap_{i\in\mathcal{N}}\mathcal{A}_i$ in which $\sqcap$ is the Cartesian product. We further denote the actions of all other nodes of the network except node i by a_{-i}, and $\mathbf{a} = (a_i, \mathbf{a}_{-i}) \in \boldsymbol{\mathcal{A}}$ is an action profile of the game. Given the actions of the nodes, the utility function maps the action profile of the game in $\boldsymbol{\mathcal{A}}$ to a real value for every user $U_i(a_i, \mathbf{a}_{-i}) : \boldsymbol{\mathcal{A}} \to \mathbb{R}$, $\forall i \in \mathcal{N}$. The game is then formally defined by the tuple $\langle \mathcal{N}, \boldsymbol{\mathcal{A}}, U\rangle$ in which the action of a node is defined as

$$a_i = \underset{a_i\in\mathcal{A}_i}{\operatorname{argmax}} \quad U_i(a_i, \mathbf{a}_{-i}), \qquad \forall i \in \mathcal{N}.$$

Game design[1] is an important branch in game theory. The goal of game design is to design the rules of the game in a way that a rational node in the game would act in favor of the objective that the system designer aims to optimize [35]. Approaches to game design for distributed optimization and network formation are given in [36] and [34]. Game design includes designing a proper utility function since not every utility function results in convergence [37]. In fact, since the nodes are selfish and act independently, their decision for their own utility maximization may not necessarily lead to a fixed-point action profile in $\boldsymbol{\mathcal{A}}$, called the equilibrium.

[1]The actual term used in game theory is "mechanism design" [34]. Here, we use the term "game design" to avoid confusion with the term "mechanism."

In the scenario of Fig. 1, the design of the utility function can, for example, concentrate on the topology formation for data dissemination in wireless *ad hoc* networks [36]. In such a case, the utility design aims at minimizing the energy required for data dissemination in the whole network while maintaining the throughput. However, a proper utility design needs to be fair in terms of the node contribution to prevent free-riding [33]. While in this scenario, a network level optimization is the goal, it is also possible to further design individual utility functions at the node level. For instance, in a video streaming scenario, various network conditions map to different user quality-of-experience (QoE) [38]. We note that not only the network conditions but also the client system configuration has a significant effect on the user perceived QoE [39]. Here, a utility function can map some of the personal preferences of a user, in terms of the sought video quality and the level of the contribution to be made to the network, and optimize the network based on those parameters in order to maximize the quality-of-experience of the user [40].

Here, the utility function is designed in such a way that a maximum throughput or a minimum delay for the different nodes can be achieved. However, at the same time, a fair network resource allocation is required [33]. Such a design of a utility function which attains a Pareto-optimal allocation of resources on top of multipath scheduling while still favoring technologies with low latency is provided, e.g., in [41].

III. MAPPING TO THE INTERNET MODEL

Next, we describe a mapping of the generalization of transitions to the five-layer Internet model to achieve a protocol-independent future Internet. Transitions allow communication systems to exchange mechanisms to retain functionality. As described above, the configuration space of a transition-enabled communication system is specified in a context feature model (comprising system and context features). A mapping of the context feature model to a layering model like the Internet model is depicted schematically in Fig. 4. Each mechanism within a multimechanism, i.e., the set of functionally equivalent mechanisms, realizes the same functionality but exhibits different performance characteristics for a given context. An example of a multimechanism on the network layer is routing. Mechanisms of this multimechanism are, for example, the Open Shortest Path First (OSPF) protocol and the Optimized Link State Routing (OLSR) protocol for mobile *ad hoc* networks. A particular mechanism implementation can often be further configured by parameters at runtime. For example, in OSPF, the route cost calculation can be adapted at runtime on the basis of link-cost parameters. Configuration parameters of mechanisms are captured by additional features and feature attributes. The choice between multiple mechanisms of a multimechanism can, as a result, be specified as an XOR-group where the multimechanism is the parent feature and the mechanisms are the corresponding children features. If multiple mechanisms of a multimechanism can be active at the same time, the corresponding multimechanism is specified as an OR-group.

The performance of a mechanism depends on the current context and other selected mechanisms. Constraints imposed by the current context are represented as cross-tree constraints between features of the context branch and features representing multimechanisms or particular mechanisms. For example, the choice of available routing mechanisms can be restricted if there is heavy traffic in the network. Multiple mechanisms on different layers that are active at the same time may introduce additional cross-layer dependencies. To consider functional cross-layer dependencies (i.e., dependencies between multimechanisms), cross-tree constraints between those features that represent multimechanisms are introduced. For example, the routing protocol OLSR requires AdHoc to be active. Further approaches additionally allow for considering non-functional cross-layer influences on configuration options (e.g., the quality of communication) [42].

The state of the transition-enabled communication system corresponds to the currently present configuration of the entire context feature model. Transitions are triggered by changing context conditions that are detected, e.g., by a dedicated monitoring mechanism. These changing context conditions are represented by reassigning features of the context branch of the context feature model. Fig. 4 depicts transitions as represented in the context feature model by selecting appropriate mechanisms within the XOR-group that represents a multimechanism. The figure also depicts self-transitions, which correspond to a reconfiguration of parameter settings of a particular mechanism without replacing the mechanism itself (e.g., adapting the link-cost parameters in OSPF). Finally, we denote by the multimechanism adaptation the change from one valid configuration of the context feature model to another valid configuration.

For an optimal behavior, the multimechanism adaptation is performed by following a transition plan as discussed in Section II-C. By careful choice of the utility function, desired adaptation behavior can then be achieved. By solving the MDP that governs the time evolution of the configuration space, an optimal transition plan is determined. The resulting multimechanism transition plan optimizes the system utility by achieving the maximum expected discounted performance. In Fig. 4, the abstraction of the adaptation of the feature diagram of the Internet model is captured by a stochastic state evolution of an MDP following the transition plan.

IV. CAPABILITIES AND LIMITATIONS OF TRANSITION-BASED SYSTEMS

By enabling transitions between functionally similar mechanisms, a transition-based system can adapt to dynamic contextual situations, satisfying a wide range of application requirements. For instance, it has been shown

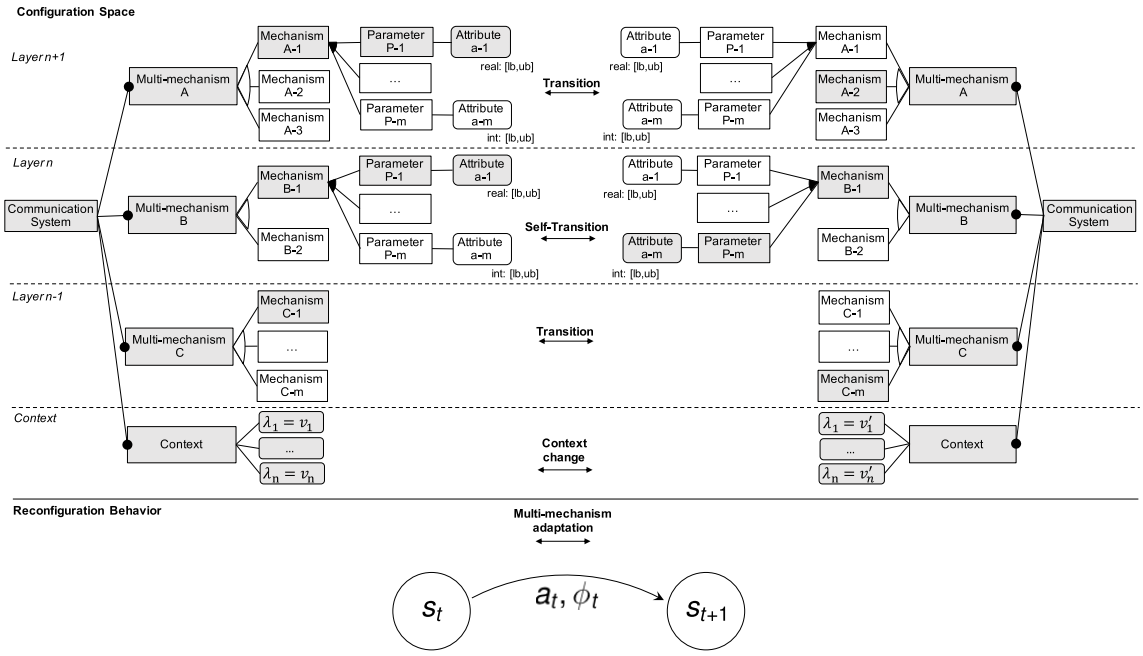

Fig. 4. ***Transition-enabled communication system. The configuration space shows the time evolution of the transition-enabled system as a feature diagram in the Internet model. The reconfiguration behavior is captured by the time evolution of an MDP. The transition plan/policy of the MDP selects mechanisms according to the action $a_t = \pi(s_t)$ and receives the performance ϕ_t. In the depicted context feature model example, this plan leads to a transition of Mechanisms A-1 to A-2, a transition of Mechanisms C-1 to C-m, and a parameter adaptation of Mechanism B-1, which we denote as a self-transition. Note that the stochastic context change is captured by the new context set λ.***

for a publish-subscribe system that the mechanism transition concept can aid in adapting to application-specific workloads and mobility characteristics [43]. Another line of work utilizes the concept of transitions in the context of distributed complex event processing systems, wherein a dynamic exchange of placement mechanisms allows fulfilling distinct performance objectives of a large number of users [44].

Context feature models enable to specify known contextual situations in regard to known system characteristics. However, the derivation of a system reconfiguration based on a feature model: 1) induces computational costs and 2) utilizes the available memory. To tackle these drawbacks, computational complexity can be shifted from runtime to design time. To this end, context feature models incorporate context information and enable system designers to reason about potential (anticipated) contextual changes. This provides the possibility to determine reconfigurations with minimum costs regarding the estimated long-term changes for a particular context [45].

The specifications described above can be derived given that context changes that require transitions are anticipated at design time. Context changes in dynamic communication systems are, however, tied to uncertainty. To this end, the MDP representation of transitions allows to incorporate uncertain context changes. Nevertheless, it is still an open issue how to address context feature model specifications that may evolve at runtime, incorporating or relaxing so far unobserved dependencies between features or even incorporating new system characteristics in terms of additional features. Although preliminary work on evolution of DSPL exists [46], this is still an open question.

The characterization of the configuration space in the previous sections takes a local perspective of a network node and implicitly assumes that other nodes that deploy different mechanisms are encoded into the context. However, it is often desirable to specify the configuration space from a global system perspective, where dependencies and variability between multiple instances of nodes and their particular characteristics are specified. To this end, more expressive specifications such as Cardinality-based feature models [47] can be employed. A configuration of a CFM consists of multiple instances of a particular feature type, which allows representing, e.g., multiple nodes of a communication system with identical configuration options. However, the considerably increased expressiveness of CFM results in a potentially unbounded number of configurations, which makes reasoning about specifications not to mention their analysis challenging [48].

One main difficulty arising with the presented formalization of mechanism transitions is that the configuration state space suffers from an exponential growth with an increasing number of contexts, features, and actions. Therefore, the computational complexity required for a smooth functioning of a transition-based system might be prohibitive, which makes approximate solutions useful alternatives. Since the available memory on the system nodes also poses a constraint, especially for processes with large state space, it is often pragmatic to allocate more resources to compute optimal policies for states that are

more frequently encountered [49]. Model reduction techniques such as state-aggregation pose a different approach to the problem of large state spaces. A traditional state-aggregation technique first identifies configurations that are similar in some sense and then lumps them together, yielding a smaller state space. One can then design an MDP similar to Section II on the reduced state space. Note that arbitrarily lumped processes may not be Markovian [50] which calls for aggregation techniques ensuring approximate Markovianness [51]. These techniques can be utilized for proving near-optimality of transitions in communication systems with a large state space.

V. CONCEPTS FOR MODELING RECONFIGURABLE SYSTEMS

Next, we highlight related concepts in the field of modeling reconfigurable systems. It is structured along three categories: state space modeling, system behavior, and applications.

The state space category is concerned with modeling the possible states of a system. Here, architectural models or feature models can be used for system modeling [52]. Both model types can be used to specify the state space of a system. Architectural models consist of components representing the (possible) composition of a software system. One well-known approach using this model type is the Rainbow framework [53], which consists of an architecture layer and a system layer with a corresponding mapping. The architecture layer incorporates the adaptation logic while the system layer represents the runtime system. The architecture layer uses constraints to check if the current model is valid or violates a system objective. In this case, an adaptation engine plans some action that gets executed by translating the plan for the system using the aforementioned mapping. Rainbow focuses on the software architecture of the managed resource. Thus, it does not explicitly model the context of the system. Feature modeling provides a different approach for modeling the system state space. For example, the FUSION framework for building self-adaptive software systems [54] proposes an approach based on online learning for determining the influence of adaptation decisions on nonfunctional goals in addition to allowing fine-tuning the system parameters at runtime. Here too, the authors do not model the context, which prevents modeling dependencies between features and attributes.

Further DSPL-based approaches include the idea of tackling design-time uncertainty of context dynamics [46] by allowing the runtime evolution of the specifications. Thus, additional flexibility at runtime is added as the specification can evolve. While changing the DSPL specification at runtime already helps when dealing with uncertainty, adding time aspects to a specification can improve adaptation decisions as well. As an example, an approach for extending DSPLs with temporal constraints to model the reconfiguration life cycle of a DSPL is found in [55]. This allows adding constraints to the transition from one configuration to another. Furthermore, an extension of DSPL, called automatic software product line (ASPL), realizes self-adaptable products [56] by intertwining automatic computing and self-adaptive software systems. This approach uses automatic element abstraction, which also implements an MAPE knowledge loop, for the ASPL architecture. The approach, however, lacks an MDP-based learning approach for optimizing the adaptation decisions under uncertainty.

System Behavior can be modeled using state automata and Unified Modeling Language (UML) state machines [52]. One approach is to add UML state machine models to each system component for specifying its adaptation behavior [57]. For orchestration, a container, which contains the state machine models, triggers the adaptations and makes sure that they are consistent. In [58], an incremental quantitative verification method is introduced for specifying and checking properties of a system model based on an MDP formulation. The method is based on decomposing the MDP into its strongly connected components in order to exploit the model structure effectively. In addition to the possibility of specifying the adaptation behavior at design time, a system can be enabled to learn a behavior at runtime using reinforcement learning techniques. MDPs are used for modeling reinforcement learning scenarios, which can be used in autonomic computing [59]. Further formal methods including MDP formulations in self-adaptive software systems are discussed thoroughly in [60] and Lemos2013SE.

Finally, we present applications of modeling reconfigurable systems that are related to communication networks. An overview of such applications is given, e.g., in [62]. The emergence of MDPs as a modern decision tool in the context of communications networks can be traced back to the initial considerations of controlled communication networks. Queuing theory provides a useful abstraction for this purpose. Expectedly, a significant amount of research effort was spent on controlled queues and their performance analysis. The works in [63]–[65] document early developments in that direction that, for example, focus on two fundamental areas of communication networks: network routing and multiple access control [64]. For example, the dynamic routing problem is formalized as a special case of an MDP which is reformulated as a linear program for optimization. Similarly, the random access problem is abstracted using a controlled Markov chain formulation with the packet retransmission probability as the control variable.

MDPs have also found interesting applications in the domain of congestion control, especially in the context of dynamic networks [66]. Here, a multiagent reinforcement learning approach enables devising a robust cooperative multiagent congestion controller that can regulate the data traffic source flows adaptively under changing environments. Interestingly, a TCP pacing protocol, called TCP

Contention Control that avoids TCP burstiness and aims at maximizing the overall throughput is formalized using an MDP [67]. Similarly, the MDP framework can be used to model bandwidth adaptation problems for the purpose of QoS provisioning in adaptive mobile communication networks [68].

Another area where MDPs have been successfully applied is scheduling in wireless networks, especially opportunistic scheduling [69], [70]. In [71], downlink wireless scheduling with a hybrid automatic-repeat-request protocol is analyzed using an MDP formulation where an optimal draining convex scheduling algorithm is proposed. For overlay networks, the problem of multipath data transfer has been cast as a Markov decision problem in [72], where a so-called online policy iteration (OPI) is proposed to solve the decision problem at runtime. The MDP-based algorithm OPI is empirically shown to achieve better performance than traditional JSQ and weighted round robin schemes.

VI. CONCLUSION

In communication scenarios of the future Internet, where the exchange of data is subject to high connectivity dynamics and stringent quality requirements, transitions become a fundamental approach to adaptive communications. Communication mechanisms that are adapted using transitions include diverse system features such as wireless interfaces, scheduling methods, and TC algorithms, to deal with scenarios of high dynamics and tight quality requirements including autonomous driving, connected industry automation, and IoT applications.

This paper introduced a formalization of the concept of transitions, which is a method to express the adaptivity of exchanging communication mechanisms at runtime. This formalization is based on integrating the description of communication system features with a decision-making model under uncertainty and methods for utility function design. Since communication mechanism combinations are constrained by system properties and context-dependent requirements, the configuration space can be abstracted by applying the methodology of DSPLs. Following this model, we showed in this paper how to map the system transitions to a transition plan, which is found by solving an MDPs. Using a Markov model of the system, the constraints of the DSPL method and the stochastic time evolution of the environment can be described.

The solution to the MDP provides a transition plan, which anticipates future context changes. Hence, the system adaptation behavior can be controlled by the design of utility functions that model each communication mechanism's benefits and costs. Intended collaborative behavior of several nodes within the distributed communication system can then be controlled using tools from game theory.

Most importantly, this paper shows that the formalization of a transition-enabled communication system can intuitively be mapped to the Internet model, where the transition logic of the system is captured by the MAPE principle. Hence, we argue that a protocol-independent deployment of applications that take advantage of all available communication mechanisms is made feasible by transitions. As detailed in this paper, the presented concept addresses the requirements of communication scenarios of the future Internet, strengthening our vision that the concept of transitions will fundamentally contribute to a protocol-independent view of the future Internet. ■

REFERENCES

[1] A. Frömmgen *et al.*, "Mechanism transitions: A new paradigm for a highly adaptive Internet," Dept. Elect. Eng. Inf. Technol., Techn. Univ. Darmstadt, Darmstadt, Germany, Tech. Rep. TUD-CS-2016-0167, Mar. 2016. [Online]. Available: http://tuprints.ulb.tu-darmstadt.de/5370/

[2] D. Liu *et al.*, "User association in 5G networks: A survey and an outlook," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1018–1044, 2nd Quart. 2016.

[3] M. Boban, A. Kousaridas, K. Manolakis, J. Eichinger, and W. Xu. (2017). "Use cases, requirements, and design considerations for 5G V2X." [Online]. Available: http://arxiv.org/abs/1712.01754

[4] Y. Brun *et al.*, "Engineering self-adaptive systems through feedback loops," in *Software Engineering for Self-Adaptive Systems*. Springer, 2009, pp. 48–70.

[5] R. Capilla, J. Bosch, P. Trinidad, A. R. Cortés, and M. Hinchey, "An overview of dynamic software product line architectures and techniques: Observations from research and industry," *J. Syst. Softw.*, vol. 91, pp. 3–23, May 2014. doi: 10.1016/j.jss.2013.12.038.

[6] N. Bencomo, P. Sawyer, G. S. Blair, and P. Grace, "Dynamically adaptive systems are product lines too: Using model-driven techniques to capture dynamic variability of adaptive systems," in *Proc. Int. Workshop Dyn. Softw. Product Lines (DSPL)*, 2008, pp. 23–32. doi: 10.1109/SPLC.2008.69.

[7] S. Hallsteinsen, E. Stav, A. Solberg, and J. Floch, "Using product line techniques to build adaptive systems," in *Proc. 10th Int. Conf. Softw. Product Lines*, Aug. 2006, pp. 141–150. doi: 10.1109/SPLINE.2006.1691586.

[8] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2008.

[9] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," Softw. Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-90-TR-021, 1990. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm? AssetID=11231

[10] K. Czarnecki, T. Bednasch, P. Unger, and U. W. Eisenecker, "Generative programming for embedded software: An industrial experience report," in *Proc. ACM SIGPLAN/SIGSOFT Conf. Generat. Program. Compon. Eng.*, Oct. 2002, pp. 156–172. doi: 10.1007/3-540-45821-2_10.

[11] D. Benavides, P. T. Martín-Arroyo, and A. R. Cortés, "Automated reasoning on feature models," in *Proc. 17th Int. Conf. Adv. Inf. Syst. Eng. (CAiSE)*, Jun. 2005, pp. 491–503. doi: 10.1007/11431855_34.

[12] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM Comput. Surv.*, vol. 37, no. 2, pp. 164–194, Jun. 2005. doi: 10.1145/1089733.1089736.

[13] T. Stahlbuhk, B. Shrader, and E. Modiano, "Topology control for wireless networks with highly-directional antennas," in *Proc. Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw.*, New York, NY, USA, May 2016, pp. 1–8. doi: 10.1109/WIOPT.2016.7492931.

[14] R. Kluge, M. Stein, G. Varró, A. Schürr, M. Hollick, and M. Mühlhäuser, "A systematic approach to constructing families of incremental topology control algorithms using graph transformation," in *Software & Systems Modeling*, pp. 1–41, 2017. doi: 10.1007/s10270-017-0587-8.

[15] I. Schweizer, M. Wagner, D. Bradler, M. Mühlhäuser, and T. Strufe, "kTC—Robust and adaptive wireless ad-hoc topology control," in *Proc. Int. Conf. Comput. Commun. Netw.*, New York, NY, USA, Jul./Aug. 2012, pp. 1–9. doi: 10.1109/ICCCN.2012.6289318.

[16] X. Chu and H. Sethu, "Cooperative topology control with adaptation for improved lifetime in wireless ad hoc networks," in *Proc. IEEE Int. Conf. Comput. Commun.*, New York, NY, USA, Mar. 2012, pp. 262–270. doi: 10.1109/INFCOM.2012.6195667.

[17] L. Li, J. Y. Halpern, P. Bahl, Y.-M. Wang, and R. Wattenhofer, "Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks," in *Proc. ACM Symp. Principles Distrib. Comput.*, New York, NY, USA, 2001, pp. 264–273. doi: 10.1145/383962.384043.

[18] X.-Y. Li, W.-Z. Song, and W. Wang, "A unified energy-efficient topology for unicast and broadcast," in *Proc. Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, 2005, pp. 1–15. [Online].

Available: http://doi.org/10.1145/1080829.1080831
[19] Y. Xing, J.-H. Hwang, U. Çetintemel, and S. Zdonik, "Providing resiliency to load variations in distributed stream processing," in *Proc. 32nd Int. Conf. Very Large Data Bases (VLDB)*, 2006, pp. 775–786.
[20] U. Srivastava, K. Munagala, and J. Widom, "Operator placement for in-network stream query processing," in *Proc. 24th ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst. (PODS)*, New York, NY, USA, 2005, pp. 250–258.
[21] H. Hartmann and T. Trew, "Using feature diagrams with context variability to model multiple product lines for software supply chains," in *Proc. 12th Int. Conf. Softw. Product Lines*, Sep. 2008, pp. 12–21. doi: 10.1109/SPLC.2008.15.
[22] A. Feldmann and W. Whitt, "Fitting mixtures of exponentials to long-tail distributions to analyze network performance models," *Perform. Eval.*, vol. 31, nos. 3–4, pp. 245–279, 1998.
[23] R. E. Bellman, *Dynamic Programming*. New York, NY, USA: Dover, 1957.
[24] R. A. Howard, *Dynamic Programming and Markov Processes*. Hoboken, NJ, USA: Wiley, 1964.
[25] A. S. Manne, "Linear programming and sequential decisions," *Manage. Sci.*, vol. 6, no. 3, pp. 259–267, 1960.
[26] M. Toussaint and A. Storkey, "Probabilistic inference for solving discrete and continuous state markov decision processes," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 945–952.
[27] W. R. KhudaBukhsh, A. Rizk, A. Frömmgen, and H. Koeppl, "Optimizing stochastic scheduling in fork-join queueing models: Bounds and applications," in *Proc. IEEE Int. Conf. Comput. Commun.*, May 2017, pp. 1–9.
[28] W. R. KhudaBukhsh, B. Alt, S. Kar, A. Rizk, and H. Koeppl, "Collaborative uploading in heterogeneous networks: Optimal and adaptive strategies," in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2018, pp. 1–9.
[29] A. Frömmgen *et al.*, "A programming model for application-defined multipath TCP scheduling," in *Proc. ACM/IFIP/USENIX Middleware*, vol. 1, 2017, pp. 134–146.
[30] M. Ghavamzadeh *et al.*, "Bayesian reinforcement learning: A survey," *Found. Trends Mach. Learn.*, vol. 8, nos. 5–6, pp. 359–483, 2015.
[31] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1. Cambridge, MA, USA: MIT Press, 1998.
[32] K. Jain and M. Mahdian, "Cost sharing," in *Algorithmic Game Theory*, N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, Eds. New York, NY, USA: Cambridge Univ. Press, 2007, pp. 385–410.
[33] M. Mousavi, S. Müller, H. Al-Shatri, B. Freisleben, and A. Klein, "Multi-hop data dissemination with selfish nodes: Optimal decision and fair cost allocation based on the Shapley value," in *Proc. IEEE Int. Conf. Commun.*, May 2016, pp. 1–6.
[34] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, Eds., *Algorithmic Game Theory*. New York, NY, USA: Cambridge Univ. Press, 2007.
[35] N. Li and J. R. Marden, "Designing games for distributed optimization," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 2, pp. 230–242, Apr. 2013.
[36] M. Mousavi, H. Al-Shatri, M. Wichtlhuber, D. Hausheer, and A. Klein, "Energy-efficient data dissemination in Ad Hoc networks: Mechanism design with potential game," in *Proc. IEEE 12th Int. Symp. Wireless Commun. Syst.*, Aug. 2015, pp. 616–620.
[37] J. R. Marden and A. Wierman, "Overcoming the limitations of utility design for multiagent systems," *IEEE Trans. Autom. Control*, vol. 58, no. 6, pp. 1402–1415, Jun. 2013.
[38] K. Brunnström *et al.*, "Qualinet white paper on definitions of quality of experience," Tech. Rep., 2013.
[39] D. Stohr, A. Frömmgen, A. Rizk, M. Zink, R. Steinmetz, and W. Effelsberg, "Where are the sweet spots? A systematic approach to reproducible DASH player comparisons," in *Proc. ACM Multimedia*, vol. 1, 2017, pp. 1113–1121.
[40] M. Mousavi, H. Al-Shatri, W. R. KhudaBukhsh, H. Koeppl, and A. Klein, "Cross-layer QoE-based incentive mechanism for video streaming in multi-hop wireless networks," in *Proc. IEEE 86th Veh. Techn. Conf.*, Sep. 2017, pp. 1–7.
[41] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. Le Boudec, "MPTCP is not Pareto-optimal: Performance issues and a possible solution," in *Proc. 8th Int. Conf. Emerg. Netw. Experim. Technol.*, 2012, pp. 1–12.
[42] M. Weckesser, R. Kluge, M. Pfannemüller, M. Matthé, A. Schürr, and C. Becker, "Optimal reconfiguration of dynamic software product lines based on performance-influence models," in *Proc. 22nd Int. Conf. Syst. Softw. Product Line (SPLC)*, Gothenburg, Sweden, vol. 1, Sep. 2018, pp. 98–109.
[43] B. Richerzhagen, "Mechanism transitions in publish/subscribe systems," Ph.D. dissertation, Tech. Univ. Darmstadt, Darmstadt, Germany, 2018. [Online]. Available: http://tuprints.ulb.tu-darmstadt.de/6669/
[44] M. Luthra, B. Koldehofe, P. Weisenburger, G. Salvaneschi, and R. Arif, "TCEP: Adapting to dynamic user environments by enabling transitions between operator placement mechanisms," in *Proc. 12th ACM Int. Conf. Distrib. Event-Based Syst.*, 2018, pp. 136–147. doi: 10.1145/3210284.3210292.
[45] K. Saller, "Model-based runtime adaptation of resource constrained devices," Ph.D. dissertation, Dept. Elect. Eng. Inf. Technol., Techn. Univ. Darmstadt, Darmstadt, Germany, Jan. 2015. [Online]. Available: http://tubiblio.ulb.tu-darmstadt.de/70574/
[46] A. M. Sharifloo, A. Metzger, C. Quinton, L. Baresi, and K. Pohl, "Learning and evolution in dynamic software product lines," in *Proc. IEEE/ACM 11th Int. Symp Softw. Eng. Adapt. Self-Manage. Syst.*, May 2016, pp. 158–164.
[47] K. Czarnecki, S. Helsen, and U. Eisenecker, "Formalizing cardinality-based feature models and their specialization," in *Softw. Process, Improvement Pract.*, vol. 10, no. 1, pp. 7–29, 2005.
[48] M. Weckesser, M. Lochau, T. Schnabel, B. Richerzhagen, and A. Schürr, "Mind the gap! Automated anomaly detection for potentially unbounded cardinality-based feature models," in *Proc. 19th Int. Conf. Fundam. Approaches Softw. Eng.*, 2016, pp. 158–175. doi: 10.1007/978-3-662-49665-7_10.
[49] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.
[50] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*, vol. 356. Princeton, NJ, USA: van Nostrand, 1960.
[51] W. R. KhudaBukhsh, A. Auddy, Y. Disser, and H. Koeppl. (2018). "Approximate lumpability for markovian agent-based models using local symmetries." [Online]. Available: http://arxiv.org/abs/1804.00910
[52] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker, "A survey on engineering approaches for self-adaptive systems," *Pervas. Mobile Comput.*, vol. 17, pp. 184–206, Feb. 2015.
[53] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-based self-adaptation with reusable infrastructure," *Computer*, vol. 37, no. 10, pp. 46–54, Oct. 2004.
[54] A. Elkhodary, N. Esfahani, and S. Malek, "FUSION: A framework for engineering self-tuning self-adaptive software systems," in *Proc. 18th ACM SIGSOFT Int. Symp. Found. Softw. Eng. (FSE)*, 2010, pp. 7–16. doi: 10.1145/1882291.1882296.
[55] G. Sousa, W. Rudametkin, and L. Duchien, "Extending dynamic software product lines with temporal constraints," in *Proc. 12th IEEE/ACM Int. Symp. Softw. Eng. Adapt. Self-Manag. Syst.*, May 2017, pp. 129–139. doi: 10.1109/SEAMS.2017.6.
[56] N. Abbas, "Towards autonomic software product lines," in *Proc. 15th Int. Softw. Product Line Conf. (SPLC)*, 2011, pp. 44:1–44:8. doi: 10.1145/2019136.2019187.
[57] C. Ballagny, N. Hameurlain, and F. Barbier, "MOCAS: A state-based component model for self-adaptation," in *Proc. 3rd IEEE Int. Conf. Self-Adapt. Self-Organizing Syst. (SASO)*, Sep. 2009, pp. 206–215.
[58] M. Kwiatkowska, D. Parker, and H. Qu, "Incremental quantitative verification for Markov decision processes," in *Proc. IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw.*, Jun. 2011, pp. 359–370.
[59] M. Amoui, M. Salehie, S. Mirarab, and L. Tahvildari, "Adaptive action selection in autonomic software using reinforcement learning," in *Proc. 4th Int. Conf. Autonomic Auto. Syst.*, Mar. 2008, pp. 175–181.
[60] D. Weyns, M. U. Iftikhar, D. G. de la Iglesia, and T. Ahmad, "A survey of formal methods in self-adaptive systems," in *Proc. 5th Int. C* Conf. Comput. Sci. Softw. Eng. (C3S2E)*, 2012, pp. 67–79. doi: 10.1145/2347583.2347592.
[61] R. de Lemos, H. Giese, H. A. Müller, and M. Shaw, Eds., *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, Oct. 24–29, 2010 Revised Selected and Invited Papers*. Berlin, Germany: Springer, 2013.
[62] E. Altman, "Applications of markov decision processes in communication networks," in *Handbook of Markov Decision Processes*. Springer, 2002, pp. 489–536.
[63] S. Stidham, Jr., and R. Weber, "A survey of Markov decision models for control of networks of queues," *Queueing Syst.*, vol. 13, nos. 1–3, pp. 291–314, 1993.
[64] A. Ephremides and S. Verdú, "Control and optimization methods in communication network problems," *IEEE Trans. Autom. Control*, vol. 34, no. 9, pp. 930–942, Sep. 1989.
[65] T. B. Crabill, D. Gross, and M. J. Magazine, "A classified bibliography of research on optimal design and control of queues," *Oper. Res.*, vol. 25, no. 2, pp. 219–232, 1977. doi: 10.1287/opre.25.2.219.
[66] K.-S. Hwang, S.-W. Tan, M.-C. Hsiao, and C.-S. Wu, "Cooperative multiagent congestion control for high-speed networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 2, pp. 255–268, Apr. 2005.
[67] H. Xie and A. Boukerche, "TCP-CC: cross-layer TCP pacing protocol by contention control on wireless networks," *Wireless Netw.*, vol. 21, no. 4, pp. 1061–1078, May 2015. doi: 10.1007/s11276-014-0833-8.
[68] Y. Fei, V. W. S. Wong, and V. C. M. Leung, "Efficient QoS provisioning for adaptive multimedia in mobile communication networks by reinforcement learning," *Mobile Netw. Appl.*, vol. 11, no. 1, pp. 101–110, Feb. 2006. doi: 10.1007/s11036-005-4464-2.
[69] A. Asadi and V. Mancuso, "A survey on opportunistic scheduling in wireless communications," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1671–1688, 4th Quart., 2012.
[70] W. Ouyang, S. Murugesan, A. Eryilmaz, and N. B. Shroff, "Exploiting channel memory for joint estimation and scheduling in downlink networks—A Whittle's indexability analysis," *IEEE Trans. Inf. Theory*, vol. 61, no. 4, pp. 1702–1719, Apr. 2015.
[71] J. Huang, R. A. Berry, and M. L. Honig, "Wireless scheduling with hybrid ARQ," *IEEE Trans. Wireless Commun.*, vol. 4, no. 6, pp. 2801–2810, Nov. 2005.
[72] V. Bui, W. Zhu, A. Botta, and A. Pescape, "A Markovian approach to multipath data transfer in overlay networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 10, pp. 1398–1411, Oct. 2010.

ABOUT THE AUTHORS

Authors' photographs and biographies not available at the time of publication.