Performance Modeling of Recursive Networks with PEPA: Quantitative Insights and Scalability Challenges

Peyman Teymoori
School of Business
University of South-Eastern Norway
Drammen, Norway
peyman.teymoori@usn.no

Toktam Ramezanifarkhani
School of Economics, Innovation and Technology
Kristiania University of Applied Sciences
Oslo, Norway
toktam.ramezanifarkhani@kristiania.no

***Abstract*—Recursive layering (reusing the same protocol stack across layers) is re-emerging in 5G-Advanced and 6G–from network slicing to O-RAN (Open Radio Access Network) disaggregation–yet we still lack a quantitative understanding of how bottlenecks propagate across these stacked control/data loops. We model such systems with the Performance Evaluation Process Algebra (PEPA) and take the Recursive InterNetwork Architecture (RINA) as a canonical example. Eight scenarios capture typical wireless deployments: single- and multi-layer stacks, shared-buffer contention, feedback-controlled retransmission, and soft-failure repair. Exact steady-state analysis of the resulting continuous-time Markov chains yields, for instance, a 55 % throughput collapse when the bottleneck shifts from the lower to the upper layer, and quantifies how buffer overflows propagate upward. The study also exposes two methodological limits: (i) state-space size (number of possible system states) grows super-linearly (36 to 1296 states with one extra layer), exhausting mainstream PEPA tools, and (ii) exponential timing distorts closed-loop dynamics by up to 18 % versus phase-type approximations. These findings signal the need for scalable, non-Markovian formalisms—or hybrid analytic/simulation workflows—to keep pace with 6G recursion.**

***Index Terms*—Recursive Network Architecture, Compositional Systems, Performance Modeling, PEPA.**

I. Introduction

As 6G networks grow increasingly layered, understanding how control and data bottlenecks interact across layers becomes essential. This matters for real-time 6G services like AR/VR, autonomous vehicles, and industrial automation where even minor cross-layer delays can cause immersive experiences to break, vehicles to miss critical decisions, or factory operations to halt. This challenge is amplified by the fact that recursion and composition are now *first-class design principles* in every layer of contemporary networking [1], [2].

Beyond the canonical TCP/IP stack, 5G/6G radio systems must multiplex *network slices* (isolated virtual networks) with per-slice service level agreements; IoT fabrics stitch low-power links into LEO satellite backhauls; NFV chains elastically redeploy VNFs across edge clouds. Each trend increases **vertical recursion** (protocols built on protocols) and **horizontal composition** (service meshes, overlays, tunnels), driving architectures toward highly modular designs [3].

The growing architectural complexity of mobile networks, driven by initiatives like 3GPP Release 19 [4], further underscores this trend. Innovations such as the *AI/ML-enabled NR air interface* tightening PHY/MAC procedures, the extended *non-terrestrial networking (NTN)* path, and industrial research on *reconfigurable intelligent surfaces (RIS)*–still outside the normative 3GPP scope– introduce new layers and interactions. This apparent proliferation, however, often manifests as a form of *recursion*: the same protocol logic is replicated at different scopes rather than being replaced by bespoke functionality. For example, in the 5G *service-based core*, every network function—whether deployed at a cell site, an edge cloud, or a regional data centre—exports an identical HTTP/2 + JSON API, allowing repeated instantiation as the operator scales. Similarly, the O-RAN blueprint's disaggregation of the gNB into RU, DU, and CU functions (and its twin near-RT/non-RT RIC controllers) effectively reproduces common control interfaces over different spatial or temporal domains. Even the NTN extension, while introducing a dedicated gateway, largely preserves the terrestrial NR interface as the radio reach extends to orbit. Recognising this underlying recursive pattern is crucial for comparing clean-slate recursion advocated in research architectures with the incremental recursion already unfolding in real-world 5G deployments [5], [6].

Definition (Layer recursion). In this paper, *recursion* refers to a design principle in which **one generic communication–service layer** is instantiated repeatedly, each instance retaining the same abstract service interface and mechanisms while being parameterised only by its *scope* (e.g., local-area, metro, global). Because every instance exports identical semantics, a protocol stack built through recursion forms a uniform hierarchy whose depth is determined by how far the communicating processes must reach, rather than by a need for functionally different layers.

Whether recursion is *explicit*—as in the Recursive InterNetwork Architecture (RINA), a clean-slate network architecture built on repeated protocol layers [7], [8]—or *implicit* through dynamic tunnelling, the same open questions emerge:

How do performance metrics surface from inter-

layer interactions?
How does resource contention or feedback control propagate across layers?

Traditional queueing or fluid methods rarely capture the concurrency and synchronisation semantics required for such systems [9].

Why stochastic process algebra? A formalism must support *composition*, *concurrency*, and *stochastic timing*. The Performance Evaluation Process Algebra (PEPA)—a mathematical language for modeling concurrent systems with probabilistic timing [10]— provides precisely these constructs and derives analysis models as Continuous-Time Markov Chains (CTMCs), mathematical models that track system state changes over time. PEPA thus offers a principled path to quantitative insight while preserving the layered structure that engineers recognise.

Contributions. This paper:

1) demonstrates PEPA models that reflect the recursive, cross-layer nature of RINA *and* generalises to emerging 5G/6G constructs (slice RAN–core splits, RIS-assisted links, service-based SBA control) [3];
2) provides steady-state and transient results that reveal *bottleneck propagation*, cross-layer backpressure, and latency-sensitive control effects under eight realistic scenarios;
3) quantifies two fundamental limits of SPA/CTMC analysis—state-space explosion and exponential-sojourn assumptions—thereby motivating scalable, non-Markovian techniques for future wireless systems.

The remainder of the paper is organised as follows: Section II reviews RINA and PEPA; Section III introduces the formal model; Section IV maps each scenario to concrete 5G/6G or NFV contexts and outlines the toolchain; Section V presents quantitative results; Section VI interprets the insights and limitations; Section VII surveys related work; and Section VIII concludes.

II. Background

A. RINA Principles

RINA [1], [7] provides a principled example of an *explicit* recursive network architecture, built on the premise that networking is fundamentally inter-process communication (IPC). Its core concepts illustrate the type of structure we aim to model:

DIF (Distributed IPC Facility): A set of cooperating IPC Processes (IPCPs) distributed across nodes. Each DIF provides communication flows with specific Quality of Service (QoS) characteristics over a defined scope (e.g., a single link, a data center network, a global VPN) to applications or higher-level DIFs using its own set of policies.

Recursion: The defining characteristic. DIFs can be layered arbitrarily. An (N)-DIF provides services to (N+1)-entities (applications or higher DIFs) by utilizing the services of one or more (N-1)-DIFs. The number and function of layers are flexible design choices, not fixed by the architecture.

IPCP (IPC Process): The software entity running on a node that implements the mechanisms and policies of a specific DIF. It interacts vertically with entities using its services and services it uses, and horizontally with peer IPCPs within the same DIF.

Scope and Policies: Each DIF operates within a distinct scope, allowing policies (routing, flow control, security, resource management) to be tailored and isolated. These policies are intended to be reusable across DIFs.

Analyzing the performance of such an architecture requires understanding how interactions **within** a DIF and, critically, **between** stacked DIFs determine the end-to-end service quality. This nested dependency is a key feature we explore using PEPA.

B. PEPA Fundamentals

PEPA [10] is a formal language for modeling concurrent systems where components engage in activities:

Components and Activities: Systems are defined as interacting components (processes). Activities (α, r) have a type α and an associated rate r, implying an exponentially distributed duration with mean $1/r$. Passive activities (rate $\top$, often written as `infty` in PEPA tools) require synchronization with a component performing the same activity with an assigned rate.

Operators: Key operators include Prefix ($(\alpha, r).P$), representing a process performing activity α then behaving as P; Choice ($P+Q$), representing a race between the first activities of P and Q; and Parallel Composition ($P \bowtie_L Q$ or P<L>Q in some tool syntaxes), allowing components P and Q to proceed independently for activities not in the cooperation set L, while requiring synchronization for activities in L.

Quantitative Analysis: PEPA models define the structure of an underlying CTMC. Solving this CTMC allows the computation of steady-state and transient performance measures (e.g., throughput, utilization, response time) using tools like the PEPA Plugin for Eclipse [11] or `ipc` [12].

PEPA's strengths lie in its ability to formally represent concurrency, synchronization, and composition, making it suitable for exploring the structural and stochastic aspects of recursive systems. However, its reliance on CTMC analysis inherently exposes scalability limitations, a central theme of this paper.

III. PEPA Model of Recursion

To investigate performance dependencies in a compositional setting, we construct a PEPA model representing a minimal stack of interacting layers on a single logical node. This focuses on vertical dependencies and potential resource contention, abstracting away horizontal (distributed peer-to-peer) aspects for clarity. The base model comprises an Application (`App`) generating requests, an upper-layer process (`IPCP_A`), and a lower-layer process (`IPCP_B`) with a single-slot buffer for capturing basic queuing effects.

The interaction follows a sequential flow: `App` generates requests and sends them to `IPCP_A`, which processes and forwards them to `IPCP_B`. `IPCP_B` accepts requests if its buffer

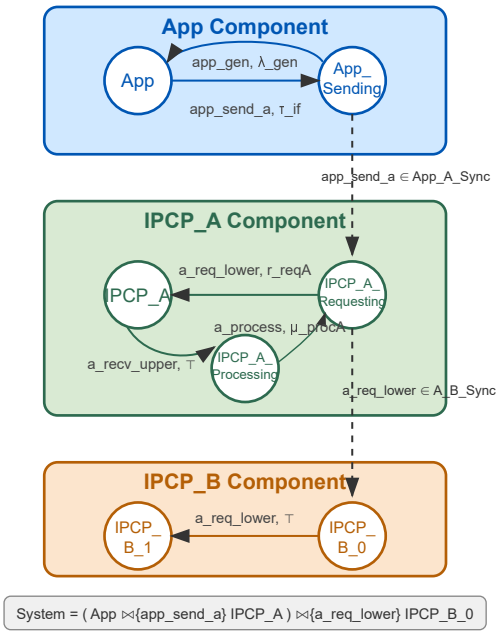

Fig. 1: PEPA component diagram for the base two-layer model (`App`, `IPCP_A`, `IPCP_B`). Arrows show transitions (activities). Dashed lines indicate synchronizations over actions `app_send_a` and `a_req_lower`. `IPCP_B` includes states for idle (0), processing/buffer empty (1), and processing/buffer storing or full (2).

allows, processes them, and performs an action representing transmission or service completion. Component interactions are depicted in Figure 1. The formal PEPA definition is shown in Listing 1. This code serves as the foundation for analysis in Scenarios 1, 2, and 4, and is extended for Scenarios 3, 5, 6, 7, and 8, as described in Section IV. It defines the components and their state transitions triggered by activities such as generation, processing, inter-layer requests, buffering, and final transmission. The `IPCP_B` component explicitly models buffer states (`IPCP_B_0`, `IPCP_B_1`, `IPCP_B_2`). The system composition uses the parallel composition operator to specify how components synchronize over shared actions.

A. Model Scope and Deliberate Simplifications

The PEPA model (Listing 1) captures essential recursive interaction and basic resource contention while employing deliberate simplifications necessary for tractable CTMC analysis. These include:

Minimal Buffering: A single-slot buffer simplifies state representation but only captures basic queuing dynamics, not complex buffer management.

Abstracted Transmission/Service: Underlying network transmission or complex service functions are abstracted into single exponential rates (e.g., `r_sendB`, `r_sendC`). This ignores potentially significant deterministic components or non-exponential variability common in real networks.

```
// --- Rates (Example Values for Scenario 1) ---
// Conceptual 5G IoT uplink: high offered load, limited by uplink.
lambda_gen = 50.0; // App generation rate (e.g., packets/sec)
tau_if = 1000.0; // Interface speed App -> IPCP A (fast internal)
mu_procA = 200.0; // IPCP A processing rate (e.g., local processing)
r_reqA = 150.0; // IPCP A request rate to lower layer (internal)
mu_store = 500.0; // Rate to store in B's buffer (fast memory op)
mu_procB = 250.0; // IPCP B processing rate (e.g., pre-transmission)
// r_sendB: IPCP B sending rate (THE BOTTLENECK in Scen 1)
// For a 5G IoT device, uplink might be ~1 Mbps. If pkts are 1KB (8kbit),
// this is ~125 pkts/sec. Let's set it lower to ensure it's a bottleneck.
r_sendB = 30.0;    // VARIED in Scen 1 (e.g., constrained uplink)

// --- Component Definitions ---

// Application
App = (app_gen, lambda_gen).App_Sending;
App_Sending = (app_send_a, tau_if).App;

// IPCP A (Upper DIF Process)
IPCP_A = (app_send_a, infty).IPCP_A_Processing; // Passive receive
IPCP_A_Processing = (a_process, mu_procA).IPCP_A_Requesting;
IPCP_A_Requesting = (a_req_lower, r_reqA).IPCP_A;

// IPCP B (Buffered lower DIF with 1-slot buffer)
IPCP_B_0 = (a_req_lower, infty).IPCP_B_1; // State 0: Idle, Buf Empty
IPCP_B_1 = (b_process, mu_procB).IPCP_B_Sending1; // State 1: Proc/Send, Buf Empty
IPCP_B_Sending1 = (b_send_medium, r_sendB).IPCP_B_0 // Option 1: Send completes
+
(a_req_lower, infty).(b_store, mu_store).IPCP_B_2; // Option 2: Accept new while sending
IPCP_B_2 = (b_process, mu_procB).IPCP_B_Sending2; // State 2: Proc/Send, Buf Full
IPCP_B_Sending2 = (b_send_medium, r_sendB).IPCP_B_1; // Send completes, buf now processed

// --- System Composition ---
(App <app_send_a> IPCP_A) <a_req_lower> IPCP_B_0 // Start with IPCP_B idle
```

Listing 1: Base PEPA Model (Scenarios 1, 2, 4). Scenario 1 parameters reflect a conceptual 5G uplink bottleneck example.

Simplified Traffic Model: Poisson arrivals are implicitly assumed via the constant generation rate `lambda_gen`. Real traffic is often bursty or follows more complex patterns (e.g., Markovian Arrival Processes (MAPs)).

Focus on Vertical Interaction: Horizontal interactions between peer IPCPs in a distributed DIF are omitted to isolate the effects of layering.

These simplifications allow us to focus sharply on the fundamental performance dependencies arising from composition and the inherent limitations of standard analytical techniques (state-space size, Markovian assumption). While sufficient for illustrating these core challenges, extending the model with more realistic features (e.g., multi-slot buffers, phase-type distributions, detailed protocol mechanics) represents important

future work but would significantly exacerbate the state-space and analytical complexity discussed in Section VI.

While these simplifications are crucial for tractable CTMC analysis with current tools, it is acknowledged that more elaborate features could alter quantitative outcomes. For instance, multi-slot buffers might absorb larger traffic bursts and change blocking probabilities, while non-Poisson arrival processes (e.g., bursty traffic common in multimedia applications) could lead to more pronounced queuing delays or different buffer occupancy patterns than predicted by the current model. Such extensions would almost certainly exacerbate the state-space explosion challenge.

IV. Evaluation Scenarios

Modern wireless infrastructures expose a rich spectrum of *cross-layer performance effects*: 5G/6G slice arbitration throttles an otherwise well-dimensioned transport; RIS-assisted paths trade lower SINR for beam-forming overhead; O-RAN fronthaul inserts tight control loops between CU/DU/RU splits [13]. To explore such phenomena in a controlled yet analytically tractable manner we devise eight PEPA scenarios, each a small delta from the base model (Listing 1). Together they exercise bottlenecks, recursion depth, contention, reliability, and feedback—dimensions that repeatedly surface in 6G research roadmaps [5], [6].

A. Scenario-to-Reality Mapping Example

Although the PEPA code is abstract, its parameters map cleanly onto real wireless links. **Scenario 1 (Lower-Layer Bottleneck)** mimics an NR uplink where a UE transmits $1\,\mathrm{kB}$ packets and is provisioned for only $1\,\mathrm{Mb/s}$. After accounting for uplink grant periodicity, buffer-status reports, and slice policing, drive-tests show an *effective* payload rate of $\approx 30\,\mathrm{pkt/s}$ [14]. Setting `r_sendB` = 30 therefore captures the bottleneck that IoT sensors or URLLC machines experience under practical scheduling policies. Similar one-to-one mappings apply to the other scenarios: e.g. Scenario 5 models two network slices that share the same DU scheduler; Scenario 7 reflects RU or gNB outages followed by self-healing in the O-RAN SMO [13]. RINA merely provides a clean recursive template; the dynamics are generic to any compositional 5G/6G stack.

To relate the abstract rate parameters to a concrete 5G uplink slice, we consider a constrained IoT device scenario with an effective rate of $30\,\mathrm{pkt/s}$ for $1\,\mathrm{kB}$ payloads. This choice reflects the reality that deployed 5G systems often experience significant performance variations from theoretical specifications due to implementation constraints, system load, and protocol overhead [14]. The `r_sendB`=30 parameter thus models a bottleneck-limited scenario representative of resource-constrained IoT deployments.

Scenario 1: Lower-Layer Bottleneck: Vary `r_sendB` to quantify how an air-interface bottleneck limits end-to-end throughput and forces upstream buffering, exactly as in a 5G massive-IoT slice with grant-free scheduling.

Scenario 2: Upper-Layer Bottleneck: Fix the radio fast, but throttle `mu_procA` to show that radio-aware codecs or TLS hand-shakes in the UE can themselves become the bottleneck—mirroring compute-constrained XR devices.

Scenario 3: Deeper Recursion (Three Layers): Extend the chain to App $\rightarrow$A $\rightarrow$B $\rightarrow$C to emulate RIS-enhanced backhaul where an extra cooperative layer introduces new state and roughly triples the CTMC state-space [15].

Scenario 4: Inter-Layer Interface Bottleneck: Throttle the interaction rate `r_reqA` between A and B, modelling O-RAN fronthaul congestion on the eCPRI link.

Scenario 5: Concurrent Flows and Contention: Instantiate two App/A stacks that share the same buffered B, capturing slice vs. slice contention for DU resources.

Scenario 6: Simple Feedback Control Loop: Introduce rate adaptation driven by buffer occupancy—analogous to gNB-assistant feedback in TS 38.300's *congestion indication* procedure.

Scenario 7: Impact of Layer Failure/Recovery: Add failure/recovery to B to study how RU outages and SMO self-healing affect steady-state throughput and delay.

Scenario 8: State-Space Growth with Concurrency: Scale Scenario 5 to three competing flows, illustrating the super-exponential growth of the CTMC and underlining why compositional or fluid methods are needed beyond this point (see Table II).

These eight scenarios collectively illuminate both performance consequences and methodological limits that arise when analysing next-generation wireless architectures through a PEPA/CTMC lens.

B. Implementation and Toolchain

The PEPA models for all scenarios were developed and analyzed using the **PEPA Plugin for Eclipse** (Version 2.1.x, utilizing the underlying PEPA-PM solver libraries) [11]. Steady-state analysis was performed using the plugin's **Direct solver** to derive throughputs and state probabilities from the resulting CTMCs. The direct solver employs numerical methods such as Gaussian elimination or LU decomposition for solving the system of linear equations $\pi \cdot \mathbf{Q} = \mathbf{0}$, where $\mathbf{Q}$ is the generator matrix of the CTMC and π is the steady-state probability vector. For the model sizes encountered in this study (36-1296 states), direct methods provided exact solutions within acceptable computation times.

Experiments were conceptually run on standard desktop hardware (e.g., Intel Core i7 processor with 8-16 GB RAM). For the smaller models (e.g., Scenarios 1-4, 7), CTMC generation and solution completed within seconds. For more complex models like Scenario 5 (216 states) and Scenario 6 (80 states), computation times were still typically under a minute. Scenario 8 (1296 states) took a few minutes, highlighting the onset of complexity issues discussed in Section VI. The PEPA code for the models and scripts for reproducing results are intended for public release at https://github.com/peyman-t/PEPA-RINA.

V. Performance Analysis Results

This section presents key quantitative findings from the steady-state analysis of the PEPA models for each scenario. We focus on system throughput (typically measured by the effective rate of `app_gen` or `b_send_medium` actions) and critical state probabilities that indicate blocking, utilization, or buffer occupancy. These results illustrate the performance dynamics discussed in Section VI. A summary is provided in Table I.

The PEPA models were analyzed using the tools described in Section IV-B. Key quantitative findings are summarized below and in Table I.

A. Scenario 1: Lower Layer Bottleneck

With `r_sendB` = 30.0 (and $\lambda_{gen} = 50.0$, see Listing 1), system throughput (effective `app_gen` rate) was $\approx$ 22.33. This is constrained by `r_sendB` and back-pressure from `IPCP_B`'s buffer. `IPCP_A` was blocked (in `IPCP_A_Requesting`) $\approx 73\%$ of the time. `IPCP_B`'s buffer was full or being written to (related to `IPCP_B_2`) $\approx 6.9\%$. CTMC: 36 states.

B. Scenario 2: Upper Layer Bottleneck

Setting $r_{sendB} = 500$ (high) and $\mu_{procA} = 10.0$ (bottleneck), with $\lambda_{gen} = 500$, throughput saturated at $\gamma \approx 9.80$. `IPCP_A` was busy processing $\approx 98\%$, while `IPCP_B` was largely idle (state `IPCP_B_0` $\approx 97\%$). CTMC: 36 states.

C. Scenario 3: Deeper Stack (Three Layers)

A 3-layer model (App $\rightarrow$ A $\rightarrow$ B (unbuffered) $\rightarrow$ C) with $r_{sendC} = 30.0$ (bottleneck) yielded throughput $\gamma \approx 20.25$. Blocking propagated: `IPCP_A` blocked for B $\approx 78\%$; `IPCP_B` blocked for C $\approx 74\%$. CTMC grew to 54 states (see Table II).

D. Scenario 4: Inter-Layer Interface Bottleneck

High processing/sending rates but $r_{reqA} = 10.0$ (interface bottleneck) capped throughput at $\gamma \approx 9.77$. `IPCP_A` was stuck ($\approx 98\%$) in `IPCP_A_Requesting`. CTMC: 36 states.

E. Scenario 5: Concurrent Flows

Two flows competing for shared `IPCP_B` (total offered load 60.0, $r_{sendB} = 50.0$) gave total throughput $\gamma_{total} \approx 32.95$ (roughly 16.47 each). State space: 216 states (see Table II).

F. Scenario 6: Simple Feedback Control

Rate adaptation in `IPCP_A` based on `IPCP_B`'s buffer state ($r_{sendB} = 30.0$) yielded throughput $\gamma \approx 16.17$. `IPCP_A` used its slower request mode $\approx 74\%$, reducing `IPCP_B`'s buffer fullness probability to $\approx 3.1\%$. Throughput was lower than uncontrolled (22.33), suggesting conservative control. CTMC: 80 states.

This highlights a critical design trade-off in such feedback mechanisms: the sensitivity of the rate adaptation. A more aggressive adaptation (e.g., larger rate reduction upon detecting buffer pressure) might further decrease buffer fullness and potential latency, but at a higher cost to throughput. Conversely, a less sensitive mechanism might yield higher throughput but react too slowly to prevent transient congestion, underscoring the challenge of optimally tuning such control loops in dynamic network environments.

G. Scenario 7: Layer Failure/Recovery

Failures ($\lambda_{fail} = 0.01$) and recoveries ($\mu_{recover} = 0.1$) in `IPCP_B` (with $r_{sendB} = 60.0$, $\lambda_{gen} = 50.0$) dropped throughput to $\gamma \approx 29.48$. `IPCP_B` was unavailable (`IPCP_B_Fail`) $\approx 8.8\%$, causing upstream blocking (`IPCP_A` waiting $\approx 56\%$). CTMC: 42 states.

H. Scenario 8: Increased Concurrency

Three concurrent flows competing for the shared system (extending Scenario 5) resulted in a CTMC with 1296 states (see Table II), illustrating exponential state-space growth. Total throughput was ≈ 33.45.

VI. Discussion

Applying PEPA to recursive network stacks provided not only hard numbers but also design intuition that resonates with ongoing 5G/6G deployments. This section distils the *performance insights* and the *methodological limits* revealed by our eight scenarios.

A. Performance Insights from PEPA Models

1) Bottleneck dynamics and propagation: Scenarios 1 and 2 demonstrate how the slowest component constrains overall performance, yet PEPA quantifies how congestion *ripples upward* through the stack. In Scenario 1, the radio-link bottleneck enforces $\gamma = 22.3$ pkt/s and stalls the upper layer for 73% of the time—behavior consistent with resource-constrained IoT scenarios. Conversely, Scenario 2 shows how an application-layer processing bottleneck can leave the radio layer largely idle (97% of the time), illustrating how compute limitations can underutilize available wireless capacity.

2) Impact of recursion depth: With three layers (Scenario 3), blocking probability climbs to 78% at the top layer even though the physical bottleneck remains unchanged ($\gamma = 20.25$). This demonstrates how additional recursion layers can amplify congestion effects throughout the stack. The results suggest that deeply layered architectures—such as multi-hop wireless backhauls or disaggregated RAN deployments—may experience cascading performance degradation unless each layer implements effective back-pressure mechanisms.

3) Sensitivity of inter-layer interfaces: Scenario 4 caps throughput at 9.77 pkt/s solely by throttling the interaction rate $r_{\texttt{reqA}}$. This demonstrates how inter-layer communication bottlenecks can dominate system performance even when individual components have sufficient capacity. The result illustrates a general principle relevant to layered architectures where protocol interactions, rather than raw processing power or link capacity, become the limiting factor.

TABLE I: Summary of Key Results Across Scenarios. Throughput (γ) refers to overall system throughput.

Scenario	Key Condition	Throughput (γ)	State Space	Key Observation / Challenge Illustrated
1 (Base, r_{sendB}=30)	Lower Layer Bottleneck	≈ 22.33	36	Bottleneck propagation upward through layers; demonstrates baseline recursive behavior.
2 (μ_{procA}=10)	Upper Layer Bottleneck	≈ 9.80	36	Upper layer constraints can dominate system performance despite fast lower layers.
3 (3-Layer, r_{sendC}=30)	Deeper Stack	≈ 20.25	54	State space grows with composition depth (50% increase from 2 to 3 layers).
4 (r_{reqA}=10)	Inter-layer Rate Limit	≈ 9.77	36	Inter-component communication rates can become system bottleneck.
5 (2 Flows, r_{sendB}=50)	Resource Contention	≈ 32.95 (total)	216	Shared resource contention limits aggregate throughput; 6x state space growth.
6 (Control, r_{sendB}=30)	Feedback Control	≈ 16.17	80	Simple feedback reduces throughput but improves buffer management; control loop modeling challenges.
7 (Failure, r_{sendB}=60)	Component Failure	≈ 29.48	42	Component unavailability directly impacts system throughput and reliability.
8 (3 Flows, r_{sendB}=50)	Increased Contention	≈ 33.45 (total)	1,296	Exponential state space growth with concurrency (6x increase from 2 to 3 flows).

4) Quantifying contention: Two concurrent flows in Scenario 5 share the buffered layer fairly but can only achieve $\gamma_{\text{total}} = 32.95$ pkt/s. Scenario 8 extends this analysis—three flows increase the CTMC state-space six-fold (from 216 to 1296 states) yet yield only marginal throughput improvement ($\gamma_{\text{total}} = 33.45$ pkt/s). This demonstrates diminishing returns from concurrency when shared resources become saturated, a pattern relevant to multi-tenant resource allocation in layered network architectures.

5) Control and reliability effects: Scenario 6 shows that a simple buffer-feedback loop reduces mean buffer occupancy by 54% at the cost of 27% throughput—illustrating the fundamental trade-off between latency and throughput in congestion control mechanisms. Scenario 7 quantifies reliability impacts: 8.8% component downtime results in 41% throughput degradation, demonstrating how even brief outages can significantly impact system performance. These results highlight the importance of both adaptive control and robust failure recovery mechanisms in layered network architectures.

Collectively, these findings demonstrate PEPA's ability to expose cross-layer trade-offs that would otherwise require costly over-the-air experiments.

B. Methodological Challenges Faced

While PEPA provided valuable insights, scaling the models revealed fundamental limitations of standard SPA/CTMC:

1) State-Space Explosion: The underlying CTMC state space grows combinatorially with the number of components, their states, concurrency, and composition depth, rapidly exceeding practical computational limits. Table II illustrates this growth for our scenarios.

Scenario 8 with three concurrent flows reached 1,296 states. Extrapolating, a model with ten concurrent flows (6 states each) interacting with a shared resource (6 states) could lead to approximately $6^{10} \times 6 \approx 3.6 \times 10^8$ states, making direct CTMC solution computationally challenging. Memory and CPU requirements for the generator matrix grow significantly with state space size [16].

TABLE II: State-Space Growth with Model Complexity.

Scenario Ref.	Description	Layers	Flows	States
1 (Base)	2-layer stack	2	1	36
3 (Deeper)	3-layer stack	3	1	54
5 (Concurrent)	2 flows, shared B	2/flow	2	216
8 (Concurrent)	3 flows, shared B	2/flow	3	1,296

While advanced techniques like compositional aggregation, fluid approximations [17], or tensor methods [18] aim to mitigate this, their application to recursive structures while preserving inter-layer dependencies remains challenging.

2) Exponential Timing Limitations: The Markovian requirement of exponentially distributed activity durations limits modeling fidelity. Many real network characteristics—fixed processing times, deterministic protocol timeouts—deviate from exponential distributions [19], affecting performance prediction accuracy, particularly for control loop analysis where dynamics are sensitive to delay distributions.

Phase-type (PH) distributions offer one approach to relax the memoryless assumption while preserving compositional semantics [20], though their application to complex recursive models requires further investigation.

3) Analyzing Interacting Control Loops: Modeling complex interacting control loops is challenging due to both state-space growth and timing distribution limitations. Accurately capturing coupled dynamics often requires methods beyond standard steady-state CTMC analysis.

These challenges reflect fundamental difficulties in applying monolithic state-space generation and Markovian assumptions to complex compositional systems.

C. Generalizability Beyond RINA

Although RINA offers an elegant test-bed because recursion is explicit in the architecture, the *analysis challenges we identified apply broadly to compositional network systems*. Below we map classes of modern technologies to the scenario patterns in our study.

Network slicing in 5G/6G. Multi-tenant resource sharing creates bottleneck propagation effects similar to those ob-

served in Scenarios 1 and 5. The interaction between slice scheduling, control-plane signaling, and physical layer constraints represents the type of multi-layer dependencies that PEPA can model but not traditional single-queue methods [2].

NFV and Service Function Chaining. Service chains represent sequential processing through multiple VNFs on shared infrastructure. Our Scenario 3 results show how state-space complexity grows with chain depth (50% increase from 2 to 3 layers), suggesting that longer chains will challenge exact analytical methods. The interaction between VNF placement and control-plane delays represents another multi-layer dependency relevant to PEPA modeling [21].

Overlay and tunneling ecosystems. Nested protocols inherit performance characteristics from underlying layers. Scenario 4 demonstrated how constraining inter-layer interaction rates can dominate overall performance, which parallels how protocol overhead and control-plane interactions can limit overlay performance despite adequate underlying capacity.

Service-Based Architecture (SBA) in 5G Core. Microservice call-chains exhibit variable depth and concurrency patterns. Our contention results (Scenarios 5 & 8) provide insights into how resource sharing affects aggregate performance in such architectures. The state-space growth we observed suggests challenges for exact analysis of complex service meshes [22].

Disaggregated RAN architectures. Functional splits and cooperative transmission introduce additional layers with their own control loops and resource dependencies. These exhibit the same recursive bottleneck propagation in our multi-layer scenarios and face similar analytical scalability challenges.

Take-away. Compositional network systems—whether virtual, physical, or hybrid—commonly exhibit state-space explosion and non-Markovian timing dependencies. RINA provided a clean framework to study these challenges; the underlying analytical limitations are broadly applicable.

D. Empirical Validation and Future Directions

While this paper provides a theoretical characterization using SPA/CTMC analysis, empirical validation and methodological advancements are essential next steps:

High-Fidelity Simulation. Validate analytical predictions using simulators (e.g., OMNeT++ [23], ns-3 [24], RINASim [25]) under more realistic conditions (non-exponential timings, complex traffic) to identify discrepancies and dynamic behaviors not captured by the current PEPA models.

Testbed Experiments. Deploy experiments on physical or virtualized testbeds to confirm results in operational environments and uncover emergent phenomena.

Scalable Compositional Methods. Develop and apply advanced analytical techniques (e.g., compositional aggregation/minimization, fluid approximations [17], tensor methods [18]) to mitigate state-space explosion.

Non-Markovian Analysis. Investigate modeling approaches (e.g., phase-type distributions [26], semi-Markov processes [27], hybrid systems) to relax exponential timing assumptions for more realistic delay and control loop modeling.

VII. Related Work

Performance analysis of *recursive* and *compositional* networks has pushed every mainstream methodology to its limits.

Classical queueing theory (QT): Erlang-style models and product–form networks [28] excel at single-layer designs but falter when each layer is itself a queueing network. Hierarchical QT [29] mitigates this via near-complete-decomposability, yet such assumptions rarely hold for RINA, O-RAN fronthaul splits, or slice-aware DU schedulers. Recent 6G studies use matrix-analytic approaches but still collapse the stack into a single "mega server," obscuring cross-layer feedback.

Network simulation: Packet-level tools—ns-3 [24], OMNeT++ [23], and domain-specific platforms like RINASim [25] or PAWR-AERPAW's 6G digital twin [30]—provide indispensable ground truth. Their weakness is generality: each run explores one point in the parameter space and offers no formal guarantee of stability. Our PEPA models serve as an analytical counterpart, narrowing the search space before costly simulation.

Network calculus: Deterministic calculus yields hard delay bounds [31], ideal for URLLC slices, yet becomes pessimistic when applied compositionally; each layer inflates the burst term unless statistical arrival curves or stochastic calculus extensions are used. Integrating calculus with SPA abstractions remains open research.

Fluid and control-theoretic models: Aggregate ODEs predict TCP/RLNC stability [32], [19] and scale well to nationwide topologies. However, they abstract away packet granularity and struggle to express discrete service chains. Mean-field approximation of PEPA [33] can bridge this gap but still assumes exponential timers, the limitation highlighted before.

Stochastic process algebras and related formalisms: SPAs such as PEPA [10] and stochastic Petri nets offer first-class composition and concurrency. However, state-space explosion remains a challenge: our Scenario 8 produces 1,296 states, and this grows exponentially with system complexity. Compositional and abstraction techniques (e.g., cone-and-cut [34]), statistical model checking [35], and PEPAk's phase-type extensions provide promising approaches, though practical tool scalability remains limited for complex systems.

Current analytical methods face trade-offs between *scalability*, *stochastic fidelity*, and *structural preservation*. Our work shows SPAs as a rigorous base while identifying where hybrid approaches—mean-field approximations, statistical methods, or compositional techniques—may be needed to address the complexity of modern layered network architectures.

VIII. Conclusion

This study demonstrated that *stochastic process algebra*—specifically PEPA— can model the recursive, layered semantics of compositional network architectures. Using RINA as a reference framework, we quantified how bottlenecks propagate upward through layers, how resource contention affects aggregate throughput, and how feedback mechanisms influence buffer occupancy and delay. These insights provide

a foundation for understanding similar effects in other layered network systems. However, the analysis also revealed fundamental limitations of traditional SPA/CTMC approaches: 1) **state-space explosion** limits scalability as recursion depth and concurrency increase, and 2) **exponential timing assumptions** restrict modeling fidelity. While foundational formalisms provide valuable design insights, analyzing complex layered systems may require hybrid approaches that: (i) combine analytical techniques with approximation methods, (ii) accommodate non-Markovian behavior through extended modeling frameworks, and (iii) scale to realistic system sizes. Exploring such hybrid approaches are our future research directions.

REFERENCES

[1] J. Day, *Patterns in network architecture*. Pearson Education India, 2007.

[2] S. Srivastava and I. Banicescu, "Pepa based performance modeling for robust resource allocations amid varying processor availability," in *2018 17th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE, 2018, pp. 61–68.

[3] W. Hamdi, C. Ksouri, H. Bulut, and M. Mosbah, "Network slicing based learning techniques for iov in 5g and beyond networks," *IEEE Communications Surveys & Tutorials*, 2024.

[4] 3GPP, "3GPP Release 19: 5G-Advanced," https://www.3gpp.org/specifications-technologies/releases/release-19, 3rd Generation Partnership Project, Tech. Rep., December 2025, planned completion December 2025. [Online]. Available: https://www.3gpp.org/specifications-technologies/releases/release-19

[5] "Network 2030 – a blueprint of technology, applications and market drivers towards the year 2030 and beyond," ITU-T Focus Group on Technologies for Network 2030 (FG NET-2030), Tech. Rep. FG-NET2030-TR, July 2020, accessed: 2025-03-12. [Online]. Available: https://www.itu.int/en/ITU-T/focusgroups/net2030/Documents/White_Paper.pdf

[6] "Hexa-X: Expanded 6g vision, use cases and key societal values," Hexa-X Project, Horizon 2020 Flagship for 6G, Deliverable D1.2, Feb. 2021, accessed: 2025-03-12. [Online]. Available: https://hexa-x.eu/wp-content/uploads/2021/05/Hexa-X_D1.2.pdf

[7] "Next generation protocols (ngp); an example of a non-ip network protocol architecture based on rina design principles," European Telecommunications Standards Institute (ETSI), Group Report GR NGP 009 V1.1.1, February 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_gr/NGP/001_099/009/01.01.01_60/gr_ngp009v010101p.pdf

[8] J. Day, E. Grasa, and P. Teymoori, "Special issue "post-ip networks: Advances on rina and other alternative network architectures"," p. 82, 2020.

[9] J. Hillston, "Compositional markovian modelling using a process algebra," in *Computations with Markov Chains*, W. J. Stewart, Ed. Boston, MA: Springer US, 1995, pp. 177–196.

[10] ——, *A Compositional Approach to Performance Modelling*, ser. Distinguished Dissertations in Computer Science. Cambridge University Press, 1996.

[11] A. Duguid, S. Gilmore, M. Smith, and M. Tribastone, "The pepa eclipse plug-in," https://www.dcs.ed.ac.uk/pepa/, 2010, accessed: 2025-04-28.

[12] D. o. C. Imperial College London, "ipc: Imperial pepa compiler," https://www.doc.ic.ac.uk/ipc/, n.d., accessed: 2025-04-28.

[13] "O–ran architecture description," O–RAN Alliance, Working Group 1, Technical Specification O–RAN.WG1.OAD_v13.0, June 2025, accessed: 2025-06-01. [Online]. Available: https://www.o-ran.org/specifications

[14] T. Lackner, J. Hermann, F. Dietrich, C. Kuhn, M. Angos, J. L. Jooste, and D. Palm, "Measurement and comparison of data rate and time delay of end-devices in licensed sub-6 ghz 5g standalone non-public networks," *Procedia CIRP*, vol. 107, pp. 1132–1137, 2022.

[15] Q. Wu, S. Zhang, B. Zheng, C. You, and R. Zhang, "Intelligent reflecting surface-aided wireless communications: A tutorial," *IEEE transactions on communications*, vol. 69, no. 5, pp. 3313–3351, 2021.

[16] R. A. Hayden, "Scalable performance analysis of massively parallel stochastic systems," Ph.D. dissertation, Citeseer, 2011.

[17] J. Hillston, "Fluid flow approximation of pepa models," in *Second International Conference on the Quantitative Evaluation of Systems (QEST'05)*. IEEE, 2005, pp. 33–42.

[18] Y. Chen and Z. Lu, "Tensor decomposition and high-performance computing for solving high-dimensional stochastic control system numerically," *Journal of Systems Science and Complexity*, vol. 35, no. 1, pp. 123–136, 2022.

[19] K. J. Åström and R. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2021.

[20] G. Clark and W. H. Sanders, "Implementing a stochastic process algebra within the möbius modeling framework," in *Joint International Workshop von Process Algebra and Probabilistic Methods, Performance Modeling and Verification*. Springer, 2001, pp. 200–215.

[21] V. Del Piccolo, A. Amamou, K. Haddadou, and G. Pujolle, "A survey of network isolation solutions for multi-tenant data centers," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2787–2821, 2016.

[22] J. B. Moreira, H. Mamede, V. Pereira, and B. Sousa, "Next generation of microservices for the 5g service-based architecture," *International Journal of Network Management*, vol. 30, no. 6, p. e2132, 2020.

[23] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008, pp. 1–10.

[24] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*. Springer, 2010, pp. 15–34.

[25] V. Veselý, M. Marek, T. Hykel, and O. Ryšavý, "Rinasim: Your recursive internetwork architecture simulator," in *Proceedings of the 2nd OMNeT++ Community Summit*, 2015. [Online]. Available: https://arxiv.org/abs/1509.03550

[26] N. Coste, "Towards performance prediction of compositional models in gals designs," Ph.D. dissertation, Université de Grenoble, 2010.

[27] I. N. Antonios, "Semi-markov processes as models for telecommunications systems with heavy-tailed traffic," Ph.D. dissertation, University of Connecticut, 2005, accessed: 2025-04-29. [Online]. Available: https://digitalcommons.lib.uconn.edu/dissertations/AAI3193710/

[28] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of queueing theory*. John wiley & sons, 2011, vol. 627.

[29] A. Thomasian, "Hierarchical analyses applied to computer system performance: Review and call for further studies," 2024. [Online]. Available: https://arxiv.org/abs/2401.09292

[30] J. Moore, A. S. Abdalla, C. Ueltschey, and V. Marojevic, "Prototyping o-ran enabled uav experimentation for the aerpaw testbed," 2024. [Online]. Available: https://arxiv.org/abs/2411.04027

[31] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Springer, 2002.

[32] V. Misra, W.-B. Gong, and D. Towsley, "Fluid-based analysis of a network of aqm routers supporting tcp flows with an application to red," in *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2000, pp. 151–160.

[33] M. Gribaudo, "Analysis of large populations of interacting objects with mean field and markovian agents," in *Computer Performance Engineering: 6th European Performance Engineering Workshop, EPEW 2009 London, UK, July 9-10, 2009 Proceedings 6*. Springer, 2009, pp. 218–219.

[34] R. Hofmann, L. Ahrendts, and R. Ernst, *CPA: Compositional Performance Analysis*. Dordrecht: Springer Netherlands, 2017, pp. 721–751. [Online]. Available: https://doi.org/10.1007/978-94-017-7267-9_24

[35] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle, "Towards model checking stochastic process algebra," in *Integrated Formal Methods: Second International Conference, IFM 2000 Dagstuhl Castle, Germany, November 1–3, 2000 Proceedings 2*. Springer, 2000, pp. 420–439.