Jamming-Resilient Handover Triggering for Programmable 6G Radio Access Networks using Reinforcement Learning

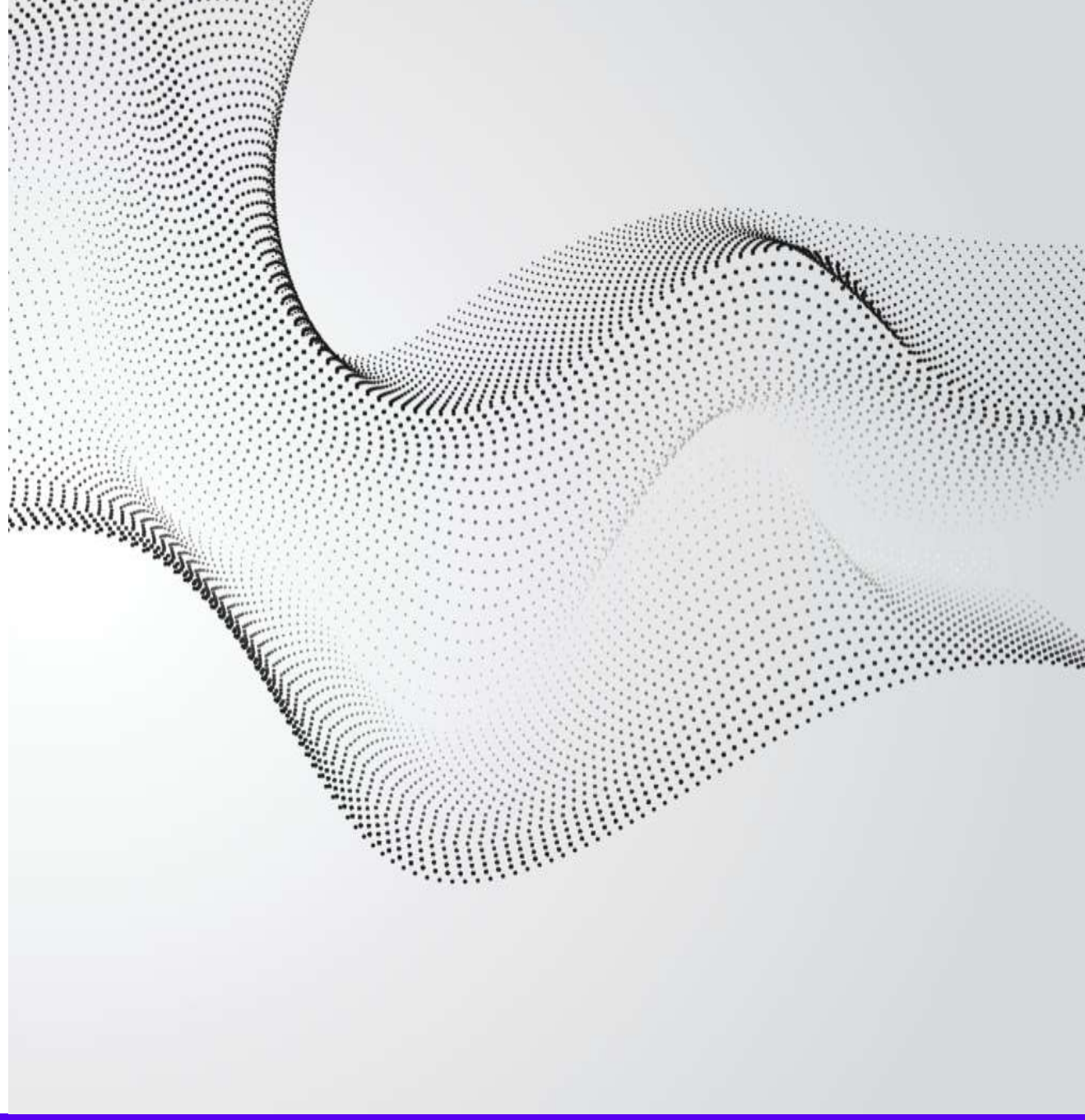
G. Amponis, P. Radoglou-Grammatikis, A. Sarigiannidis,
G. Kakamoukas, T. Boufikos, T. Lagkas, V. Argyriou,
T. Kollatou and P. Sarigiannidis

Presenter: Thomas Boufikos

tboufikos@k3y.bg

Outline

- Introduction & Motivation
- Our contribution
- Background Knowledge
- The proposed RL-based policy mechanism
- Architecture
- Implementation & Evaluation
- Conclusion & Future Work



Introduction & Motivation

Resilient operation under sudden Radio Frequency disruption is critical for the next-generation cellular networks.

- Conventional **static event-trigger handover** logic creates a fatal race condition under jamming.
- Slow-moving metrics like Reference Signal Received Power (**RSRP**) arrive **too late to be decoded** successfully by a user (UE).
- The Signal to Interference Noise Ratio (**SINR**) of UE **collapses rapidly** due to jamming attack, resulting in **radio link failure**.

Issue	Impact
Jamming attack + static event-trigger handover	Fatal race condition under jamming
RSSP metric arrives too late to be decoded	SINR of a UE decreases quickly and radio link is terminated

Contribution

The basic contribution of this paper can be summarized through the following bullet points:

- RL-based policy that **replaces the traditional fixed threshold rule for handovers** between a serving and an adjacent base station.
- The RL agent is encapsulated as an **O-RAN near-real-time xApp** and examines a reduced state vector for learning the signs of a jamming attack and **proactively completing the handover** action.
- The **3GPP protocols** themselves remain **intact** in our implementation.

Background: 5G/6G terminology

➢ Handovers

- In any telecom environment, base stations allow UEs to ***report the signal quality of both the serving cell and the neighbor cells*** using multiple metrics like RSRP and SINR.
- A handover between two cells is conducted, where an ***A3 event*** occurs.
- According to 3GPP specifications, an A3 event occurs, when the ***neighbor cell becomes offset better than the serving cell for a preconfigured Time-To-Trigger (TTT) interval***.
- This is an "intrinsic vulnerability" of 3GPP design regarding the mobility triggers, since incorrect settings (values) of offset or TTT can result in either ***ping-pong handovers or radio links failure***.

➢ RIC

- RIC stands for RAN Intelligent Controller and is a key component of O-RAN
- It ***optimizes and manages the RAN using AI, automation an programmability***, improving efficiency and performance.
- The RIC is divided into two logical components: (a) real-time RIC and (b) non-real-time RIC

Background: 5G/6G terminology

➢ Real-time RIC and xApp

- Time scale from ***10ms to 1s***.
- Runs closer to the network (***edge***) for making ***fast decisions***, such as the adjustment of beamforming parameters, the dynamic allocation of radio resources and the interference management.
- Communicates with gNB via the ***E2*** interface and with non-real-time RIC via the ***A1*** interface.
- Hosts ***xApps***, applications that are able to react rapidly to network events.

➢ Non-real-time RIC and rApp

- Time scale ***above 1s***.
- Runs on ***cloud*** or a central data server, in the Service Management and Orchestration (SMO) layer for making ***strategic decisions***, such as long-term optimization, policy enforcement and network analytics.
- Communicates with gNB via the real-time RIC via the ***A1*** interface.
- Hosts ***rApps***, applications that can optimize RAN through AI-driven insights.

Background: Reinforcement Learning and Q-Learning

➢ Reinforcement Learning (RL)

- RL is a distinct ***branch of ML***, where an ***agent*** learns to make decisions by interacting with its environment in order to maximize a ***reward function***.
- We can imagine reward function as a signal that ***quantifies how good or bad an action/outcome*** is expected to be.
- The overall concept is similar to a ***feedback loop system***, since the agent acts inside an environment, the environments responds with a reward, the agent updates its strategy and finally, it learns how to act in each state.

➢ Q-Learning

- Q-Learning is a ***specific RL algorithm***, where the agent learns the Quality of (state-action) pairs using a Q-table.
- A Q-table stores entries corresponding to each ***(state, action) pair*** and contains an ***estimation of a long-term reward.***
- Over time, the values of the Q-table ***converge to their optimal*** value, so now the agent is able to take the proper action when it is found in a current state, by knowing the ***expected reward***, found in the Q-table.

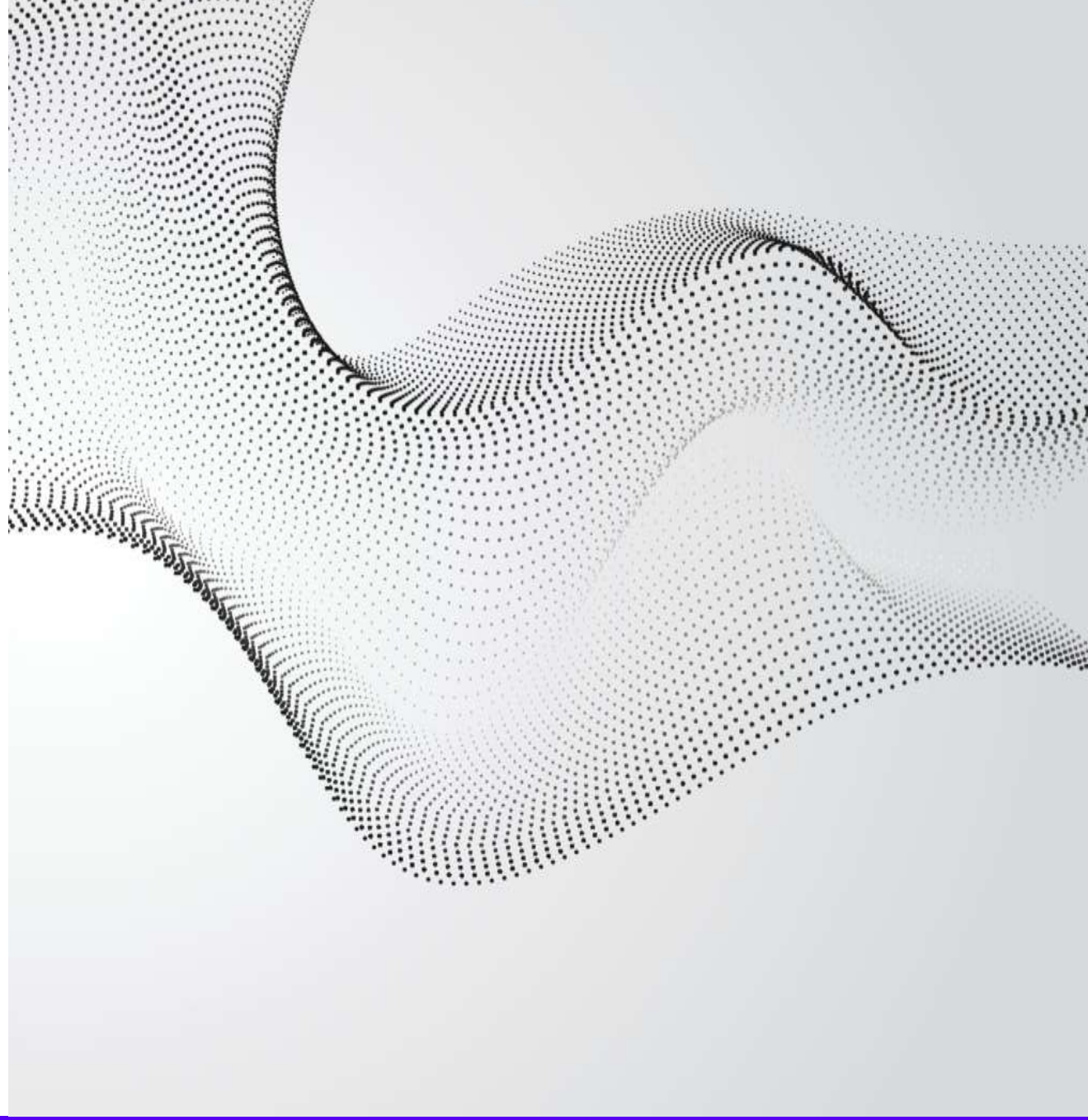
The proposed RL-based policy mechanism

❑ The problem

- Reactive ***jammers can collapse the SINR*** of a serving link within a few milliseconds, before any recovery action is completed.
- No handover takes place during this time frame, since the RSSP metric arrives too late.
- The ***serving link becomes unusable*** and even if there are any operational neighbor cells, the communication is terminated.

❑ The solution

- A lightweight, ***RL-based agent*** that interprets real-time KPIs to issue traditional handover commands or dynamic updates to UE's cluster.



The proposed RL-based policy mechanism

➢ RL-based agent vs RIC vs 6G

- RIC can host AI-driven defense mechanisms that react to PHY layer threats in just a few milliseconds.
- The RL-based agent can be implemented in ***Python and represents a RIC-hosted xApp*** that responds to jamming-induced handover triggers.
- This is in accordance to 6G's vision of zero-trust and self-defending networks.

➢ Agent's contribution

- Replaces the traditional static rule with a ***proactive, security-oriented policy***.
- *How? It* ***triggers handovers preemptively*** within the critical time window, when the radio link is still alive and the network can ***successfully deliver the handover command***.

Architecture framework

➢ The overall architecture

In order to respond to wideband jammers and trigger handovers proactively when needed, we define the following functional prototype architecture:

1. Python RL-based = xApp
2. TCP socket = E2 interface between xApp and gNB
3. C++ hook = E2 agent

➢ The agent's architecture

The agent builds a reduced state vector s, which is essentially a three-tuple with discretized components.

$$s = \{SINR_{bin}, \Delta SSR_{bin}, NACK_{bin}\}$$

Where:

1. $\boldsymbol{SINR_{bin}}$: the discretized $SINR$
2. $\boldsymbol{\Delta RSRP_{bin}}$: the RSRP difference between the serving and the neighbor gNB
3. $\boldsymbol{NACK_{bin}}$: the density of HARQ NACKs (NACK means that the receiver detected errors in the received data and requests a retransmission)

Architecture framework

Since there are 3 possible values for discretized SINR, 2 for serving-to-neighbor difference and 3 for HARQ NACK density, it is understood that there are ***18 possible states***.

The three metrics serve as ***recognizable KPIs that indicate if the agent should trigger a proactive handover***. In any case, since the agent will either trigger a handover or not, the action space is minimal and defined as:

$$A = \{defer, trigger\}$$

Regarding the reward signal r of the reward function, we can see the right-side matrix for the ***values mapping***.

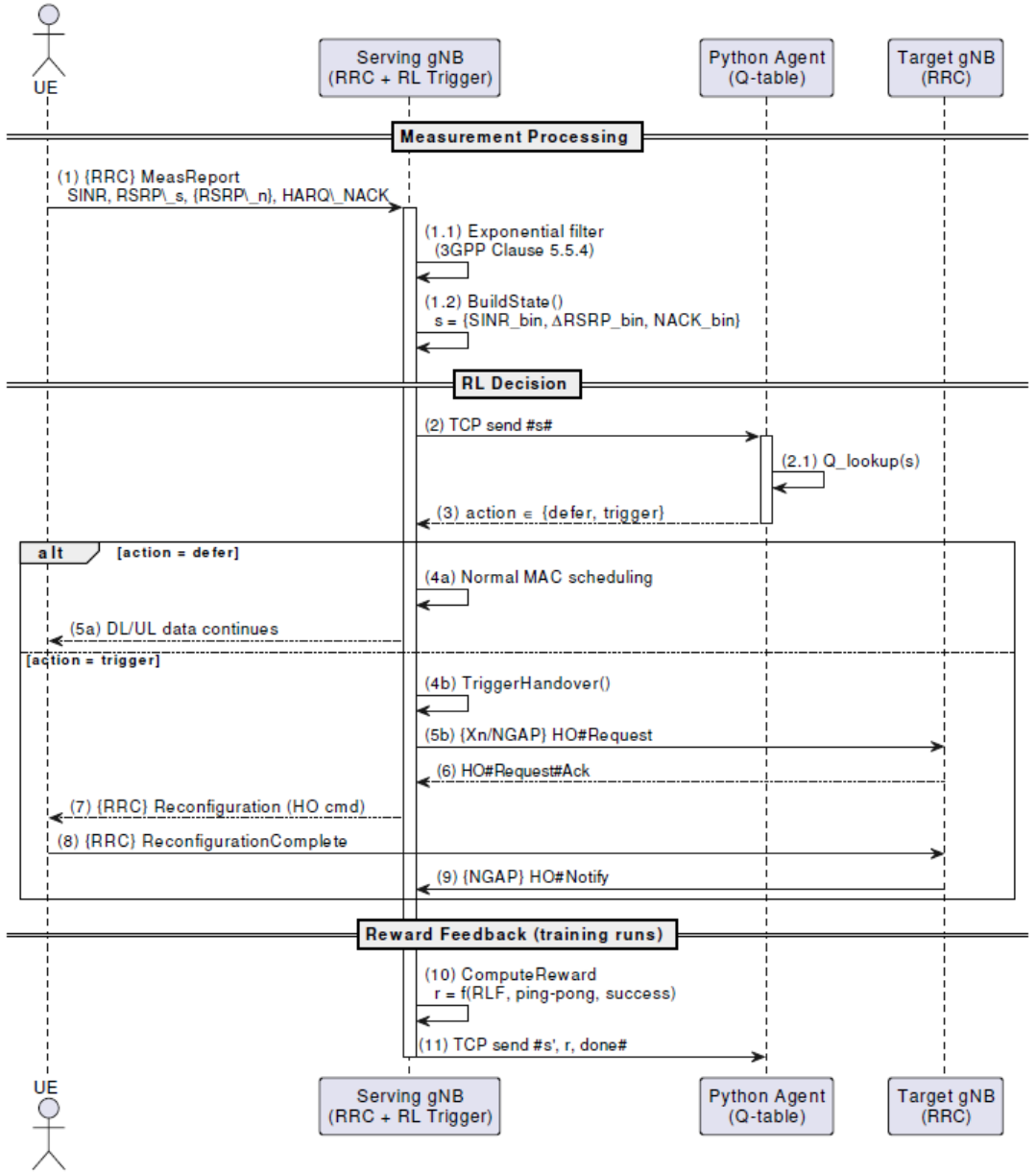
Metric	Level	Meaning
$SINR_{bin}$	Low	< 0 dB
	Medium	0 – 10 dB
	High	> 10 dB
ΔSSR_{bin}	Negative	Serving cell is stronger
	Positive	Neighbor cell is stronger
$NACK_{bin}$	Low	< 10%
	Medium	10% – 40%
	High	> 40%

Value r	Impact	Meaning
-10	Radio Link Failure	Large penalty: connectivity lost
-5	Ping-pong HO	Moderate penalty: instable network
+1	Healthy defer action	Small reward: Discourage unnecessary HOs
+10	Successful stable HO	Small reward: Proactive HO

Sequence Diagram

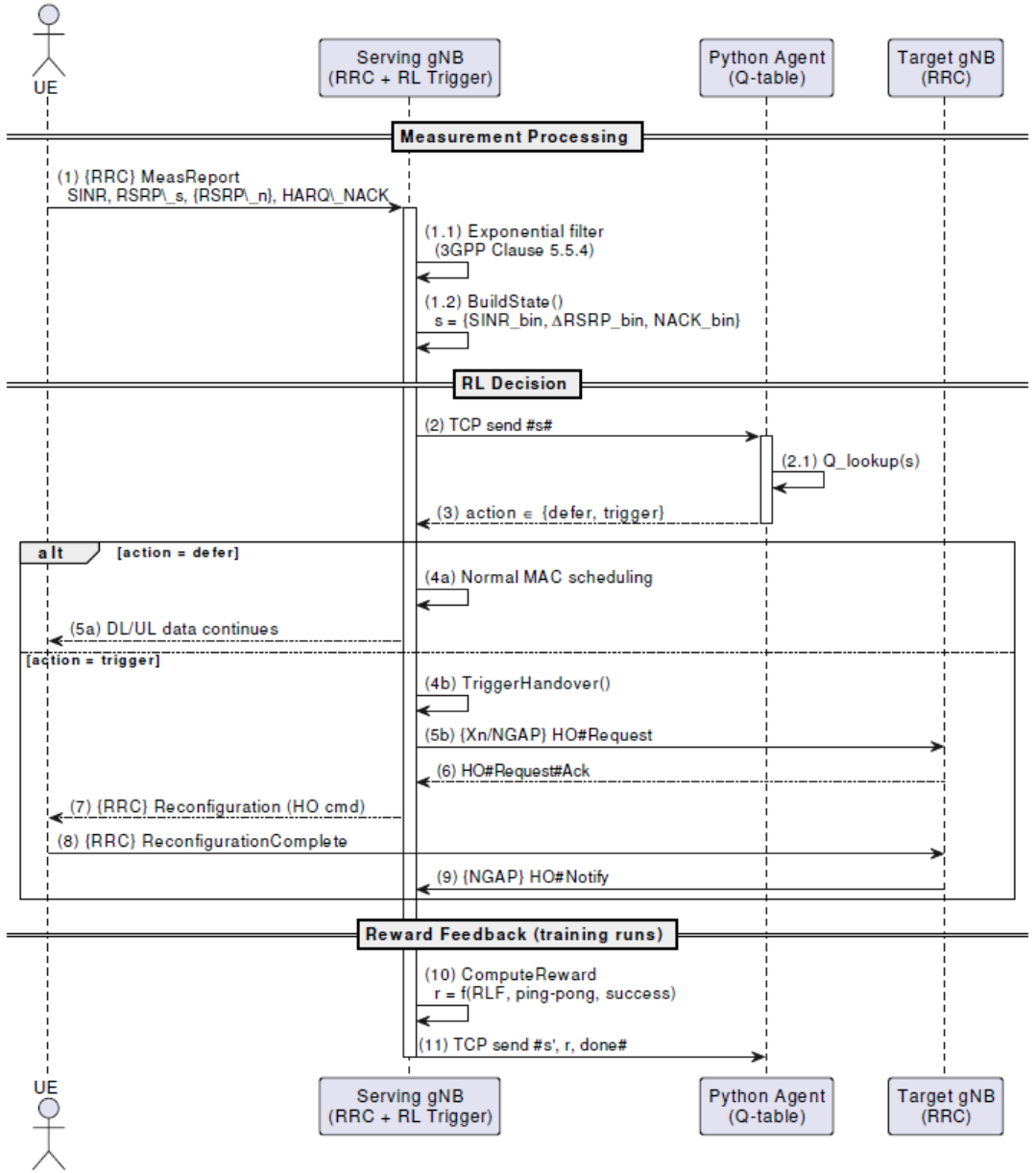
In the diagram, there are ***4 actors***: (a) the UE, (b) the serving gNB, (c) the Python agent which uses the Q-table and (d) the target gNB.

1. The serving gNB receives one of the $RRC\_MeasReport$ periodic messages from the connected UE. The message contains the values of $RSRP$ and $SINR$ and the ***C++ hook intercepts the report and build the state vector s***.
2. Through a TCP socket, the vector s is ***sent to the Python agent***.
3. The agent uses its ***Q-table to perform a lookup*** on the received state and responds back to serving gNB with its action *($defer$* or *trigger)*
4. If *action == defer*, gNB continues with normal MAC scheduling.



Sequence Diagram

4. If *action == trigger*, the standard **3GPP handover** procedure will take place.
5. A *Xn/NGAP* ***HO request*** is sent from serving gNB to the target gNB via the Xn/NGAP interface.
6. The target gNB responds with a *Xn/NGAP* ***HO request ACK***.
7. The serving gNB uses the RRC protocol to issue an ***RRC Reconfiguration*** command towards the UE.
8. Finally, the **handover is completed** and an *RRC Reconfiguration Complete* message is sent from UE to the target gNB.
9. The serving gNB **gets notified** for the successful handover event with an *NGAP HO Notify* message originating from target gNB.
10. The serving gNB ***observes the action's outcome*** (e.g., successful HO, ping-pong event, radio link failure) and ***computes the reward r***.
11. The serving gNB ***sends the reward r and new the state vector s'*** back to the Python agent, which then updates its Q-table.



Implementation & Evaluation

In the following diagrams, two baselines will be used for comparison to our proposed framework:

1. **Optimized Static A3**: An exhaustive sweep over A3-Event's *offset = {0,1,2,3,4,5,6} dB* and *TTT = {0,1,…,320} ms* – selected the combination that minimizes handover failures under the specific wideband noise profile.
2. **Reactive Power Ramping:** Upon a rapid SINR drop, the UE increases its transmission power by 3 dB for 200 ms as a mitigation action against jamming.

➢ **Setup options**

- ❑ NS3-based simulation with NR/LENA component
- ❑ 4 gNBs at the 4 corners of a 1km-length rectangle
- ❑ A wideband jammer in the center of the rectangle, producing Gaussian noise
- ❑ RL: 1000 episodes and 6o seconds per episode
- ❑ Learning rate: $\alpha = 0.1$
- ❑ Discount factor: $\gamma = 0.9$
- ❑ Exploration range: *0.05 to 0.50*

Implementation & Evaluation

Figure 1: $\boldsymbol{P(HO_{fail}) = f(P_{noise})}$

- As the noise power increases, the number of failed handovers also increases for all schemes, as expected.
- However, the RL agent approach is far more robust, as it demonstrates great resilience, even under high-power noise injection ($40dbm = 10\ Watts$).

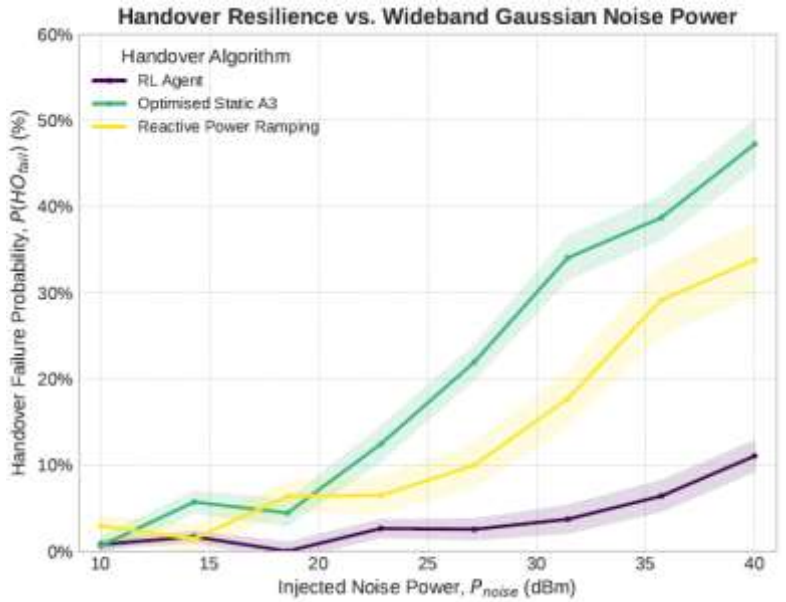
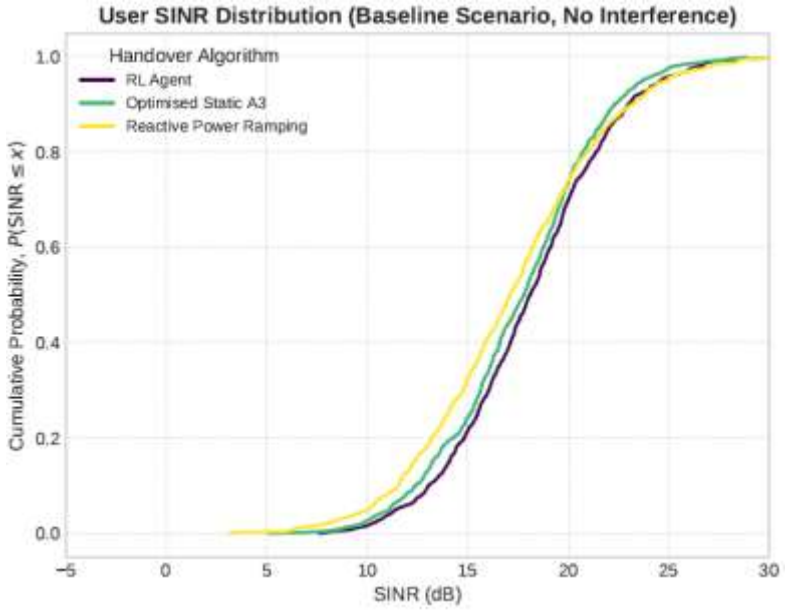


Figure 2: $\boldsymbol{CDF}$ **of** $\boldsymbol{SINR}$ **(noise only)**

- The Cumulative Density Function (CDF) of SINR has almost identical shape, regardless the deployed algorithm, in no-interference scenario.
- The RL agent does not affect the system in a clean channel environment, since all algorithms maintain a high SINR distribution.



Implementation & Evaluation

Figure 3: CDF of $SINR$ (noise + interference)

- In this case, wideband Gaussian noise has been added, so the jammer acts as an interference source.
- The RL agent proactively triggers handovers and maintains a healthier radio link for the connected UE, keeping the SINR in a range, where control commands can be successfully decoded.

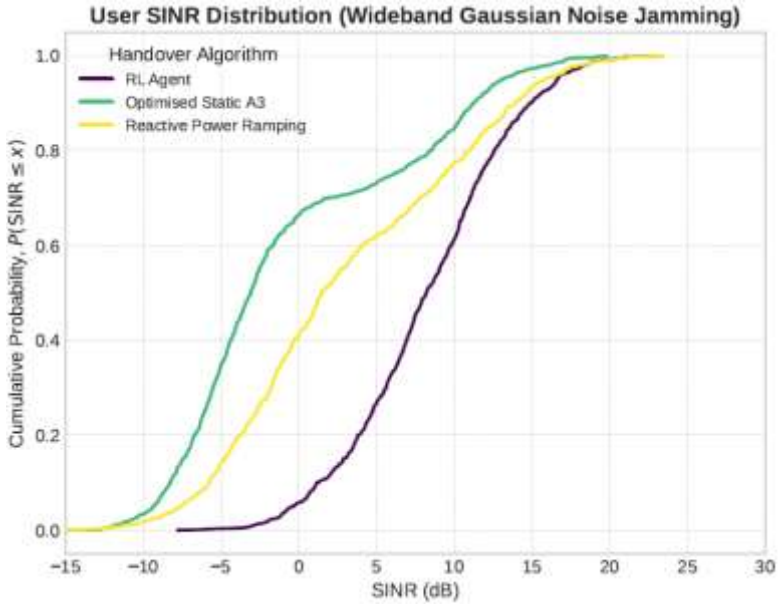
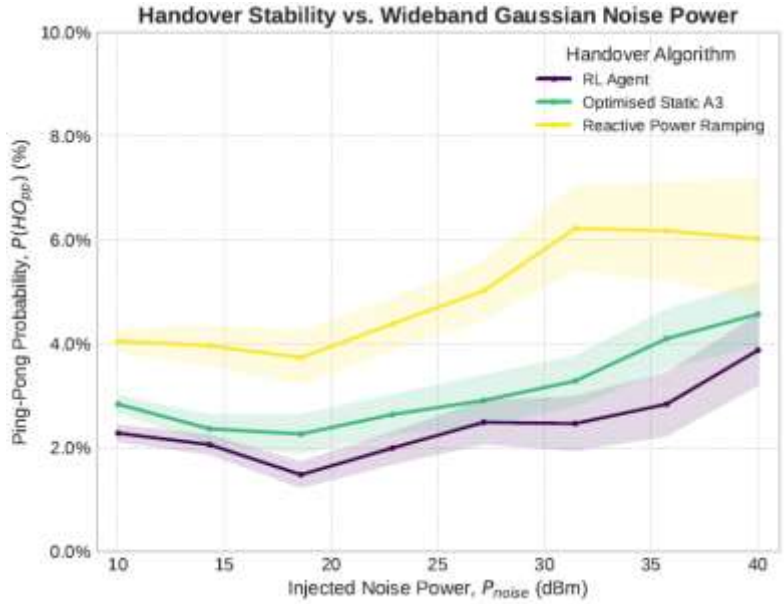


Figure 4: $P(HO_{ping-pong}) = f(P_{noise})$

- As the noise power increases, the number of ping-pong handovers typically increases for all schemes.
- Around the 15-20 dBm area, all schemes demonstrate a dip, showing a decision-freezing effect.
- However, the RL agent is again resilient without any substantial trade-off, since its stability degrades in a graceful manner, staying within the acceptable limits.



Conclusion & Future Work

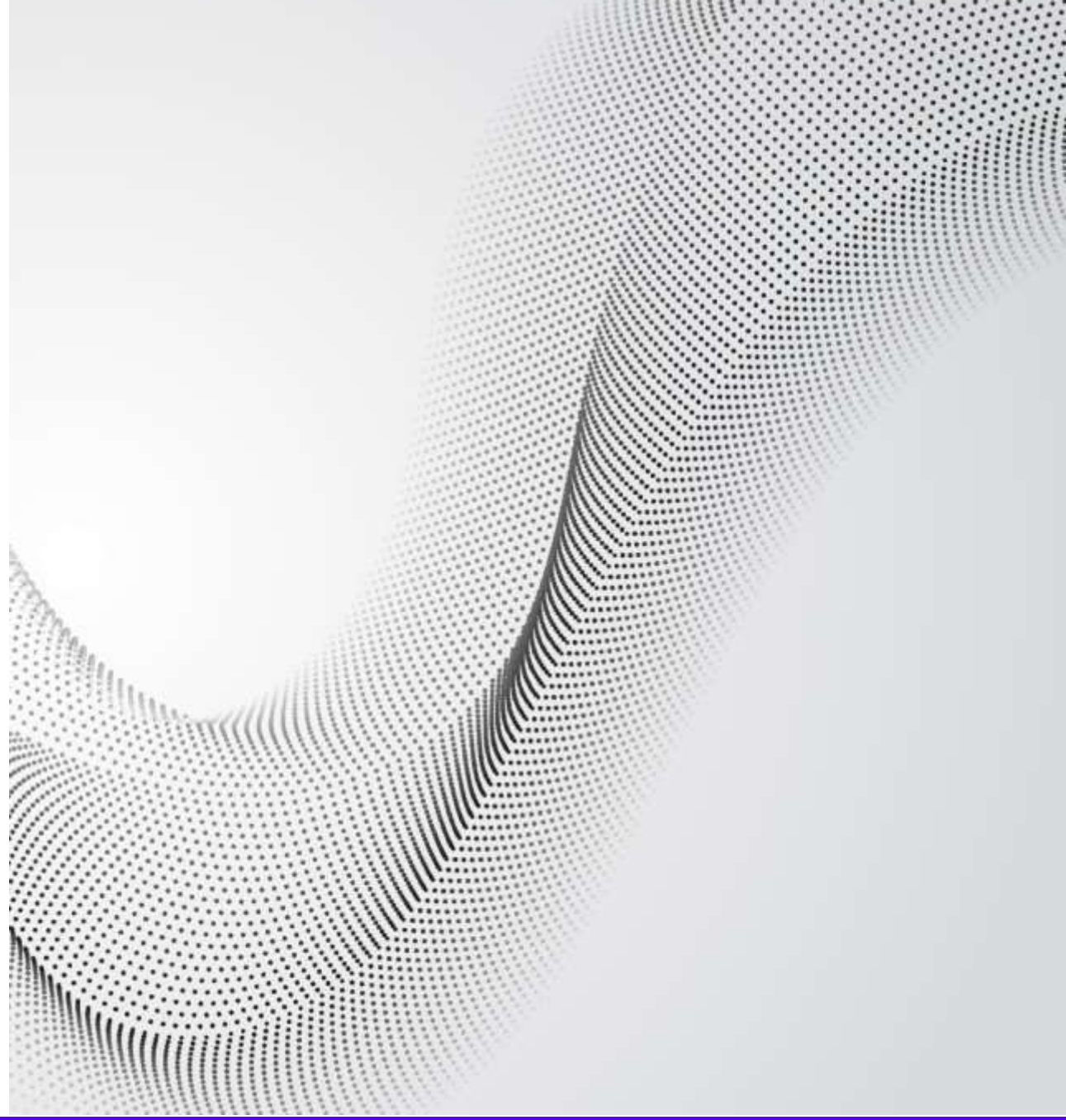
➢ Conclusion

- We replace the static 3GPP rule for handover command with a Python-based RL agent that enforces proactive policy and offers resilience and network stability.
- The agent learns to treat rapid instantaneous SINR drops and HARQ NACKs density as indicators of a potential radio link failure, before the link degrades → initiates preemptively a handover
- The agent also learns that there is a penalty for unnecessary handovers (ping-pong effect), so that it proceeds in such an option, only when a viable gNB exists.
- The system's design is an analogue to the RIC paradigm, since the Python script serves as the xApp, the TCP socket as the E2 interface and the C++ hook as the agent in E2 interface.

➢ Future Work

- The agent's state vector can be extended to include user's velocity for high mobility scenarios, cell-load metrics for network congestion scenarios, even trust scores and authentication flags for security-oriented applications.
- Scaling of the current solution, by replacing the Q-agent with a Deep Q-Network for multi-user and multi-gNB environments.

Thank you



XTRUST-6G is co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Smart Networks and Services Joint Undertaking. Neither the European Union nor the granting authority can be held responsible for them.
This work has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI)