Electronics and Information Technologies, 2026, 34, 145–164
http://publications.lnu.edu.ua/collections/index.php/electronics/index

UDC: 004.728.3:004.85

ADAPTIVE PROTOCOL SELECTION LAYER FOR CONTEXT-AWARE MULTI-PROTOCOL DATA DELIVERY IN CLOUD–EDGE SYSTEMS: A CONTEXTUAL-BANDIT FOUNDATION WITH BAYESIAN WARM-START

Roman Nazarevych* , ***Ivan Bolesta***
Department of Radiophysics and Computer Technologies,
Ivan Franko National University of Lviv
107 Gen. Tarnavsky Str., UA-79017 Lviv, Ukraine

Nazarevych, R., & Bolesta, I. (2026). Adaptive Protocol Selection Layer for Context-Aware Multi-Protocol Data Delivery in Cloud–Edge Systems: A Contextual-Bandit Foundation with Bayesian Warm-Start. *Electronics and Information Technologies*, 34, 145–164. https://doi.org/10.30970/eli.34.9

ABSTRACT

Background. Protocol performance in cloud–edge systems is regime-dependent: the transport that minimizes latency, loss, or CPU cost changes with payload, rate, link type, and signal quality, and tail latency often inverts the median ranking. Static choices and hand-tuned heuristic switchers cannot track these shifts, and many nearby adaptive-selection prototypes remain deterministic scorers, not online learners. This paper establishes the theoretical foundation for an adaptive protocol selection layer that treats protocol choice at decision ticks as online decision-making under partial feedback, demonstrated on isolated proof-of-concept examples.

Materials and Methods. Protocol selection is formulated as a contextual multi-armed bandit over the five supported implementation transports and solved with LinUCB: a per-arm ridge-regression estimator with an upper-confidence-bound exploration term, computed by Gaussian elimination. The four-dimensional context couples a log-linear payload feature, normalized Wi-Fi signal and link-rate features, and a bias term. Bounded penalty and peer-relative rewards are derived from blended median/tail latency, goodput, and loss. Non-stationarity is handled by per-tick geometric decay, and a Bayesian warm-start encodes offline ridge priors in the initial state; normal per-tick decay may scale that prior before the first logged score.

Results and Discussion. The five-protocol formulation is demonstrated on isolated Raspberry Pi 5B and Pi Zero 2 W examples; the adaptive algorithm uses five protocols (gRPC, gRPC-bidi, MQTT, CoAP, HTTP). Baseline runs confirm the best transport changes with payload, host, and objective; decision logs show LinUCB concentrating on a strong context-conditioned arm in stable regimes, while a deliberately retained cold-start miss exposes early-evidence path dependence; a Bayesian warm-start improves cold-start latency and throughput.

Conclusion. The paper gives a reproducible contextual-bandit formulation with Bayesian warm-start for multi-protocol cloud–edge data delivery. It defines context, reward, and exploration sufficient for further research.

Keywords: contextual bandit, LinUCB, adaptive protocol selection, cloud–edge systems, Bayesian prior, exploration-exploitation.

INTRODUCTION

Cloud–edge systems, including remote-instrumentation deployments such as Internet-accessible laboratories [1], deliver data over a few interchangeable application transports – gRPC [2], MQTT [3], CoAP [4], HTTP – each with different framing, session, congestion, and reliability semantics, and each evolving independently (QUIC and HTTP/3 [5, 6] are recent instances). Across cross-protocol IoT benchmarks [7], no single transport dominates across regimes: the protocol that minimizes median round-trip time at a 256-byte payload over Wi-Fi is often not the one that minimizes tail latency, loss, or edge CPU at a larger payload or on a wired link, and the ranking can invert between the 50th and 99th percentiles. The "best" protocol is thus a function of context, not a constant.

The conventional response is either a static, globally fixed protocol choice or a hand-tuned heuristic switcher with weighted scores, hysteresis, and moving averages. Both are brittle: a static choice cannot follow regime changes, and a heuristic switcher encodes its trade-offs in fixed weights and thresholds that must be re-tuned per deployment and gives no principled treatment of the exploration/exploitation dilemma – choosing the apparently best protocol now forecloses the information needed to tell whether an under-measured protocol is better. The closest prototypes in the literature remain deterministic scorers; online-learning theory, by contrast, frames this problem as regret minimization under partial (bandit) feedback [8, 9, 10, 11, 12, 13].

This paper develops the theoretical foundation for an adaptive protocol selection layer that closes this gap. We formulate protocol choice at adaptive decision ticks as a contextual multi-armed bandit and instantiate it with LinUCB [14, 15], whose decision index decomposes transparently into exploitation and uncertainty-driven exploration terms. We also derive a Bayesian warm-start that converts an offline benchmark corpus into per-arm priors, shortening the cold-start phase in which an unprimed bandit selects almost randomly. The scope is deliberate: we present the context design, reward function, exploitation/exploration mechanism, non-stationarity treatment, and prior warm-start – all reflecting a working reference implementation – and demonstrate them on isolated worked examples from a heterogeneous Raspberry Pi testbed. The systematic payload/rate evaluation, offline-replay parameter study, and reward-shaping and exploration-coefficient ablations are deferred to a companion paper; this article fixes the methodological baseline for that study.

This paper's contributions are: (i) a precise contextual-bandit formulation of multi-protocol cloud–edge data delivery with an explicitly bounded context vector; (ii) two bounded reward shapings – an absolute payload-aware penalty and a peer-relative advantage – derived from latency, loss, and goodput; (iii) a decay-based non-stationarity model integrated into the per-arm ridge estimator; (iv) an algebraic Bayesian warm-start identity with a tunable prior strength; and (v) a proof-of-concept demonstration on a heterogeneous Raspberry Pi testbed that makes the exploitation/exploration split, the safety filter, and the warm-start identity observable on real traffic. All equations map directly to the implemented controller, so the formulation is independently reproducible.

MATERIALS AND METHODS

Problem formulation

Define the action set A as the five supported transports: $A = \{gRPC, gRPC\text{-}bidi, MQTT, CoAP, HTTP\}$. At each decision tick t, the controller observes a context vector $x_t \in \mathbb{R}^4$ (defined below), selects one protocol $a_t \in A$, routes subsequent messages through it for the configured decision interval, and later observes a scalar reward r_t derived

from the observed latency, loss, and throughput of that protocol over the interval since the previous tick. Crucially, the controller observes a reward only for the protocol it selected (and for any protocol that produced resolved traffic in the window) – this is the defining partial-feedback structure of a bandit problem [9, 13]. Conceptually, this is a regret-minimization problem against the unknown context-conditioned optimal protocol rather than the maximization of a fixed weighted score, which is what motivates the bandit formulation adopted here. A quantitative regret analysis – for example, cumulative-regret curves on offline replay – is beyond the scope of this foundational paper and is deferred to the companion paper; the worked examples below instead characterize behavior through aggregate latency and throughput statistics and arm-selection shares. Decisions are pinned for a configurable interval after a switch to prevent flapping. Rewards are accumulated from an append-cursor delta so that each update reflects only the samples produced since that arm's previous update.

Context vector

The per-decision context is a four-dimensional vector that combines a workload feature, two link-quality features, and a bias term:

$$x = [payloadNorm, rssiNorm, txBitrateNorm, 1]^T. \quad (1)$$

The payload feature maps message size log-linearly onto the unit interval over the range [256 B, 1 MB] and saturates outside it (with $LN2 = \ln 2$ and the constant $12 = \log_2(1\ \text{MB} / 256\ \text{B})$):

$$payloadNorm(p) = clip\left\{\frac{\ln\left\lceil \frac{\max(p,1)}{256} \right\rceil}{LN2 \cdot 12}, 0, 1\right\}. \quad (2)$$

A logarithmic scale matches how transports experience size – a 256 B → 1 KB increase is perceptually similar to a 256 KB → 1 MB increase – and, more importantly, the explicit clip keeps the context norm $||x||$ bounded. Since the LinUCB exploration term grows with $||x||$, an unbounded payload feature would cause the exploration bonus to diverge at large payloads, and the bandit would fail to converge. Bounding the feature is done to avoid this problem. The link-quality features are sourced from a background poller of the edge device's network endpoint and normalized as:

$$rssiNorm = \text{clip}\left\{\frac{[rssi_{dBm} - (-90)]}{60}, 0, 1\right\}, \ default\ 0.5, \quad (3)$$

$$txBitrateNorm = \text{clip}\left(\frac{txMbps}{300}, 0, 1\right), \ default\ 0.5 \quad (4)$$

RSSI is mapped from [−90, −30] dBm and link rate from [0, 300] Mbps (a typical 802.11n peak; 300 Mbps is the two-stream 40 MHz 802.11n maximum). On non-WiFi links, or before the first poll completes, both default to the neutral value 0.5. The constant bias term absorbs each arm's intercept. Payloads are additionally discretized into a small number of buckets (default 8) using the same *payloadNorm* mapping; the buckets index the empirical baselines used by the reward function. We note a known degeneracy: on a

wired link *rssiNorm* and *txBitrateNorm* are held at the neutral constant 0.5, so the effective feature subspace collapses from four dimensions to two: *[payloadNorm, bias].* This does not cause underdetermination in the numerical sense – the precision matrix A_a is initialized to the identity (the ridge regularizer) and receives positive diagonal mass at every update, so the linear system $A_a z = b_a$ stays full-rank and uniquely solvable throughout. The practical consequence is that the two constant columns add no arm-differentiating signal: on a wired link, per-arm ordering reflects only payload sensitivity and the per-arm intercept, while any wired-link quality variation (interface speed, congestion, half-duplex state) stays invisible to the learner. Two of the four model degrees of freedom are therefore wasted on informationally dark dimensions. The exploration bonus $\alpha\sqrt{x^T A_a^{-1} x}$ does still converge as *payloadNorm* varies across ticks, but convergence is slower than in a full-rank context because the effective information per tick is lower. This limitation motivates the richer wired-context design planned for the companion paper.

LinUCB estimator: exploitation and exploration

Each arm *a* maintains a ridge-regression state: a 4×4 precision matrix A_a and a 4-vector b_a. A_a is the accumulated outer-product design matrix (initialized to the identity, which is the ridge regularizer [16]); b_a is the reward-weighted feature accumulator. The point estimate of the per-arm weight vector is the ridge solution:

$$\theta_a \;=\; A_a^{-1}\, b_a. \tag{5}$$

The inverse is never materialized; the linear system $A_a z = rhs$ is solved by Gaussian elimination with partial pivoting [17], an $O(d^3)$ operation that is negligible for $d = 4$ and numerically stable for the well-conditioned, identity-regularized A_a. Each protocol is scored by its upper-confidence-bound index, which is the sum of an exploitation term and an exploration term:

$$p_a(x) \;=\; \theta_a^T\, x + \alpha \cdot \sqrt{x^T\, A_a^{-1}\, x} \tag{6}$$

The first term $\theta_a^T x$ is the model's current best estimate of the expected reward for context x (pure exploitation). The second term is the one-sided confidence width: $x^T A_a^{-1} x$ is large when arm *a* has few observations in the region of feature space near x, so the bonus rewards uncertainty. The coefficient α (default 1.0) sets the exploration intensity; $\alpha \to 0$ is greedy exploitation, larger α explores more aggressively [14, 18]. At each decision tick, the candidate set is first reduced to the protocols that satisfy the viability filter (described below), which excludes any arm that is mechanically unsuitable for the current context – for example, CoAP above its supported payload boundary, or an arm currently in viability cooldown. The controller then evaluates the upper-confidence-bound index $p_a(x)$ for each remaining arm and selects the one that maximizes it:

$$a_t = \underset{a \,\in\, viable}{\operatorname{argmax}}\; p_a(x_t). \tag{7}$$

At cold start $\theta_a = 0$, for every arm, the index reduces to the exploration term alone; at the very first tick $A_a = I$, for all arms, and the shared context makes the bonus $\alpha\sqrt{x^T x}$. Identical across arms, so the initial pick is an arbitrary tie-break, not real exploration. Once arms begin to differ, the bonus favors whichever arm is least sampled near x, so the

controller explores; as evidence accumulates, the exploitation term dominates and the policy concentrates on the context-conditioned best arm. This is the principled treatment of the exploration/exploitation trade-off that fixed heuristic scorers lack.

Online update and non-stationarity

After the reward r for context x is computed for an arm, its state is updated by the standard recursive least-squares step:

$$A_a \leftarrow A_a + x \cdot x^T, \qquad b_a \leftarrow b_a + r \cdot x. \tag{8}$$

Network conditions drift, so old evidence must not dominate indefinitely. Once per active decision tick (not on pinned ticks, and not for an arm in viability cooldown), every arm is decayed geometrically toward the regularized:

$$A_a \leftarrow \gamma A_a + (1-\gamma)I, \qquad b_a \leftarrow \gamma b_a \tag{9}$$

with $\gamma \in (0, 1]$ (default 0.99). The $(1-\gamma)I$ term prevents A_a from becoming near-singular as the old signal fades, keeping the solve well-conditioned [19]. Values of γ near 1 give long memory; smaller γ adapts faster but is noisier. For arms placed in cooldown by the viability filter, the geometric decay step (Eq. 9) is suspended for the entire quarantine period. Without this exception, every active tick would shrink both A_a and b_a toward the identity prior, gradually erasing the large negative reward that triggered the gate in the first place. As θ_a relaxes back toward zero and $x^T A_a^{-1} x$ grows, the UCB index for the quarantined arm would rise purely as an artefact of forgetting, and the controller would re-select it as soon as cooldown expires – only to have it gated again. Freezing decay during cooldown preserves the evidence that justified the gate, so the arm is re-admitted based on explicit re-evaluation rather than uncertainty.

Reward function

The LinUCB selector learns from a scalar reward derived from three performance indicators – latency, message loss, and throughput – designed to be both interpretable and sufficiently discriminative across workload conditions. The simplest form is an absolute reward that compares each measured outcome against fixed reference values: latency is penalized when it exceeds a predefined budget, throughput when it falls below a target, and loss as its rate increases. This directly measures how far a protocol deviates from its expected performance targets.

However, fixed reference values are not robust across payload sizes: larger messages naturally require more transmission and processing time, so a single latency budget applied to all payloads is frequently reached or exceeded by larger ones. Because the penalty is clipped to the interval [0, 1], clearly different outcomes then collapse to the same saturated value – under a fixed 20 ms budget, latencies of 35, 60, and 120 ms all receive the maximum penalty. This saturation reduces the resolution of the reward signal, giving the bandit nearly identical feedback for protocols whose actual performance differs substantially and weakening learning in large-payload regimes. To address this, we use two reward-shaping modes: absolute and relative.

For each arm for which at least one resolved response is recorded within the delta window, an observation is formed from the windowed median (p_{50}) and tail (p_{99}) round-trip times, payload size, and loss count. The *latency signal* ℓ is defined as a fixed equal-weight combination of the two percentiles, which makes tail behavior one of the primary objectives:

$$\ell = 0.5 \cdot p_{50} + 0.5 \cdot p_{99}, \tag{10}$$

$$g = \frac{payload \cdot 1000}{\max(\ell, 1)}. \tag{11}$$

Here g is an internal goodput proxy in bytes per second, used only to rank arms inside the reward; it is distinct from the message-rate throughput (messages per second) reported in the Results. If an arm sent messages but received no responses in the window, its reward is fixed at −1 (the worst value) in both reward modes. Otherwise, one of two bounded shapings is applied.

Absolute reward mode converts a single protocol observation into a score between −1 and 0, where 0 means the protocol performed perfectly and −1 means it performed as badly as possible. The score is a penalty – the worse the protocol behaved, the more negative it gets.

To compute this penalty, the system first sets a target budget for each metric – essentially an acceptable performance threshold. For latency, the budget B_ℓ (Eq. 12) is either:

- *a learned threshold*: the protocol's recent average latency (tracked as an EWMA – a smoothed running average) multiplied by a headroom factor of 1.25, giving the protocol 25% slack above its typical performance, or
- *a cold-start fallback*: an analytic formula based on a fixed base of 15 ms plus up to 135 ms scaled by normalized payload size, used when not enough data has been collected yet.

From the observation, three individual penalty components are computed (Eq. 13):

- $latPen$ – how much the measured latency ℓ exceeds the budget B_ℓ, clipped to [0, 1]. Zero if latency is within budget; 1 if it's at or beyond the budget.
- $lossPen$ – packet loss rate q scaled by 10, clipped to [0, 1]. A loss rate of 10% or more fully saturates this component (score = 1), meaning the packet loss has a very high impact on reward.
- $thrPen$ – how far the goodput proxy g falls short of its budget B_g, clipped to [0, 1]. Zero if throughput meets the target or 1 if it falls to zero.

These three components are combined into a single reward (Eq. 14) using fixed weights: latency weight $w\ell = 0.45$, loss weight $w_q = 0.35$, goodput weight $w_g = 0.20$. Latency and loss together account for 80% of the score, reflecting that they matter more than raw throughput on constrained edge links. The division by the sum of weights is technically an identity at the defaults (they already sum to 1), but is kept explicitly so the reward stays bounded if we decide to adjust the weights.

$$B_\ell = baseMs + scaleMs \cdot payloadNorm, \;\; or \;\; B_\ell = ewma_\ell \cdot headroom, \tag{12}$$

$$\begin{aligned} latPen &= \text{clip}\left(\frac{\ell}{B_\ell}, 0, 1\right); \\ lossPen &= \text{clip}(10\, q, 0, 1); \\ thrPen &= \text{clip}\left(1 - \frac{g}{B_g}, 0, 1\right), \end{aligned} \tag{13}$$

$$r_{abs} = -\text{clip}\left(\frac{w_\ell \cdot latPen + w_q \cdot lossPen + w_g \cdot thrPen}{w_\ell + w_q + w_g},\ 0,\ 1\right). \quad (14)$$

The *EWMA* (Exponentially Weighted Moving Average) baselines used for the budgets are updated each round using (Eq. 15) with smoothing factor $\beta = 0.20$ – meaning each new observation contributes 20% to the running estimate and the previous average retains 80%.

$$m \leftarrow (1-\beta)\cdot m + \beta \cdot obs, \quad \beta = EWMA\ factor,\ default\ 0.20 \quad (15)$$

Relative reward mode uses the same observation and running averages, but instead of scoring against a predefined baseline, it scores the protocol relative to the other candidate protocols. This mode activates once at least two arms have accumulated enough observations to have stable EWMA baselines in the current payload bucket.

The comparison baselines – med_ℓ, med_g, med_q – are the medians of the ready arms EWMA values that were measured by now. They represent the typical performance level of the available protocols as seen so far.

Three advantage signals are computed (Eq. 16):

A_ℓ – log-ratio of the peer median latency to this arm's latency. Positive when this arm is faster than the median.

A_g – log-ratio of this arm's goodput proxy to the peer median. Positive when this arm has higher throughput.

A_q – linear difference between the peer median loss rate and this arm's loss rate, scaled by 10. Positive when this arm loses fewer packets.

These are combined with the same weights into a single advantage Δ (Eq. 17).

$$A_\ell = \ln\left(\frac{med_\ell}{\ell}\right);\ \ A_g = \ln\left(\frac{g}{med_g}\right);\ \ A_q = \left(med_q - q\right)\cdot 10, \quad (16)$$

$$\Delta = \frac{w_\ell \cdot A_\ell + w_q \cdot A_q + w_g \cdot A_g}{w_\ell + w_q + w_g}. \quad (17)$$

Because *Δ* is on an arbitrary scale, it is normalized (Eq. 18) by centering on the median advantage across all ready arms and dividing by a robust spread measure (MAD – median absolute deviation, floored at 0.02 to avoid division by near-zero). This makes the score comparable across different operating regimes.

$$c = \text{median}(\Delta_i);\ \ s = \max(\text{median}\ |\Delta_i - c|, minSpread);\ \ \widehat{\Delta} = \frac{\Delta - c}{s}. \quad (18)$$

The normalized advantage is then passed through a $\tanh$ function clipped to [−1, 1] (Eq. 19), with a gain of 2.0 that sharpens separation between protocols that are close in performance. Unlike the absolute reward (always ≤ 0), the relative reward can be positive; a protocol that outperforms its peers gets a positive score, which more aggressively steers the bandit toward the best option in a stable environment.

$$r_{rel} = \tanh\left(gain \cdot \hat{\Delta}\right) \quad (19)$$

Until two baselines are ready, the controller falls back to the absolute reward. The relative reward lies in [−1, 1] and can be positive when a protocol beats the current peer median, sharpening the separation between near-tied arms in a stable regime; because the baseline is online and payload-bucket-specific, relative reward values are comparable within a regime but not across unrelated regimes.

Bayesian prior warm-start

The LinUCB selector normally starts with no knowledge about which protocol is good. At the beginning of a run, all protocol arms have almost the same model parameters, so the selector must spend some time exploring all the protocols before it can learn which one performs best. This early exploration phase wastes time and may deliver poor performance in the first few runs. To reduce this cost, each protocol arm is pre-loaded with a starting model fitted from historical (non-adaptive) benchmark data. This technique – seeding a bandit from past observations – is a well-established way to shorten the exploration phase [20].

The pre-loading works as follows: for each protocol separately, a ridge regression model [16] is fitted on the historical data. Its target is an offline utility score, a single number that summarizes how well a protocol behaved in the past. This score combines three factors:

- Reliability – how often the protocol succeeded without errors
- Tail-latency sigmoid – a smooth penalty for high worst-case latency (using a sigmoid so the penalty grows gradually, not as a hard cutoff)
- CPU headroom – how much processing capacity was left unused

This offline score uses the same input features as the online model, but it is a different quantity from the online reward computed in equations (Eq. 14) and (Eq. 19) – it is only used to initialize the model, not during live operation. The target y in (Eq. 20) multiplies a reliability term $(1 - loss)$, a tail-latency sigmoid $\sigma(-p_{99}/\tau)$, and a CPU headroom term $(1 - cpu)$, and the model parameters θ are found by ridge regression with regularizer λ:

$$\min_\theta \sum_i (y_i - \theta^T x_i)^2 + \lambda\|\theta\|^2, \quad y = (1 - loss) \cdot \sigma\left(\frac{-p_{99}}{\tau}\right) \cdot (1 - cpu), \quad (20)$$

where σ is the logistic function, τ is an RTT decay scale in milliseconds, λ is the ridge regularizer, and loss and CPU are normalized fractions in [0, 1]. The prior JSON used at runtime stores fitted vectors θ_{prior} keyed by protocol, together with metadata such as d, τ_{ms}, and λ. At runtime, each arm is initialized so that its ridge state algebraically encodes the prior; the online controller loads the θ vectors and does not re-fit the offline prior. Setting the initial state to a scaled identity and a scaled prior:

$$A_a^{(0)} = s \cdot I\,, \qquad b_a^{(0)} = s \cdot \theta_{prior}. \quad (21)$$

Substituting the initial values from (Eq. 21) into the standard parameter update (Eq. 5) confirms that the model starts at exactly the prior vector:

$$\theta_a^{(0)} = (s \cdot I)^{-1} \left(s \cdot \theta_{prior}\right) = \theta_{prior}. \quad (22)$$

The initial setup encodes the prior knowledge directly into the model's starting state. Before any real data arrives, the model behaves as if it has already seen s observations consistent with the prior. The scalar s (default 1.0) controls how strongly the prior holds: a larger s means each new real observation has less relative influence, so the model takes longer to drift away from the prior. As more data accumulates, the prior's influence fades naturally.

One practical note: the per-tick decay step (which gradually reduces the weight of old observations) can run before the first real score is logged, so equations (Eq. 21) and (Eq. 22) describe the initialization state.

Finally, if a prior file is specified but fails to load, the system stops execution. This ensures a warm-start run is never accidentally treated as a cold one.

Safety and viability filter

Learning and reliability are kept separate. Before the bandit index (Eq. 6) is compared, a viability filter removes protocols that must not be selected in the current context, independently of their estimated reward: CoAP is hard-excluded for any payload strictly above a fixed 1900 B limit, an arm whose recent loss exceeds a threshold or whose recent receive count is zero (after a minimum sample count) is gated, and a gated arm enters a cooldown whose duration grows by exponential backoff on consecutive gate fires and resets after a healthy window. If every candidate is filtered out, the controller falls back to a configured safe-default protocol. This filter is a hard constraint applied to the action set before the reward calculation. Keeping the safety constraint outside the learned objective, rather than encoding it as a conservative reward penalty as in conservative-bandit formulations [21], means safety does not depend on the learner having converged.

Reference testbed

The formulation above is implemented in a working multi-protocol benchmark stack. The testbed consists of two edge devices – a Raspberry Pi 5 Model B (8 GB RAM, Gigabit Ethernet) and a Raspberry Pi Zero 2 W (512 MB RAM, 2.4 GHz Wi-Fi) – and a laptop/desktop client on Gigabit Ethernet. A deterministic offline replay harness drives the identical production selector against per-protocol isolated-baseline traces, enabling parameter studies without re-running on hardware. The selective worked examples reported below exercise this testbed only to make the formulation's moving parts observable on real traffic; the exhaustive payload/rate sweep, the replay parameter study, and the reward-shaping and exploration-coefficient sensitivity analyses are deferred to the companion paper. The software stack is implemented in Kotlin 1.9.22 (JVM 17). Transport client library versions: gRPC-Netty 1.62.2 and Protobuf 3.25.3 (gRPC and gRPC-bidi), HiveMQ MQTT Client 1.3.3 (MQTT), Eclipse Californium 3.9.0 (CoAP), and the Java built-in HTTP server from JDK 17 (HTTP).

Code and data availability

The selector implementation, the multi-protocol benchmark stack, and the decision logs used for the worked examples reported in this paper are publicly available at https://github.com/lemberh/proto-bench-pub/releases/tag/ELIT_2026Q2_publication. The repository provides the LinUCB selector, the per-protocol isolated-baseline traces, and the deterministic offline replay harness, enabling full reproduction of the reported results given the same configuration and candidate protocols. Also referenced version of the repository contains test results used in this paper and decision logs in the *bench-runner/results* directory.

RESULTS AND DISCUSSION

This paper is the foundational first part of a two-part series; its purpose is to establish the methodology and to demonstrate it on isolated worked examples. The reference testbed described above is exercised on the two heterogeneous edge devices – a Raspberry Pi 5 Model B on Gigabit Ethernet and a Raspberry Pi Zero 2 W on 2.4 GHz Wi-Fi – with a laptop client. Unless stated otherwise, the adaptive examples below use throughput mode with *throughputMaxInFlight* = 1 where applicable, LinUCB with *--banditAlpha* 0.5 (alpha=0.5), *--banditDecay* 0.99 *(decay factor* $\gamma = 0.99$*), --banditRewardMode relative*, and the five logged candidate protocols *{gRPC, gRPC-bidi, MQTT, CoAP, HTTP}*. The results below are deliberately selective: a baseline characterization that motivates the formulation, two decision-log walkthroughs that make the LinUCB index and the Bayesian warm-start observable, and an isolated-baseline reading of the selected runs. The exhaustive payload/rate sweep, the offline-replay parameter study, and the reward-shaping and α/γ sensitivity analyses are the subject of the companion paper and are not claimed here.

Baseline regime dependence

The isolated baseline results confirm that protocol performance is regime-dependent. Across the Raspberry Pi 5B and Raspberry Pi Zero 2W runs, the protocol minimizing median round-trip time is often not the protocol minimizing tail round-trip time, maximizing throughput, or minimizing edge CPU. The resulting winner grid (**Fig. 1**) changes across payloads and devices, which rules out a single globally optimal transport and motivates adaptive selection at decision ticks. These results confirm empirically what the Introduction asserts: optimal protocol choice is context-dependent, and a context-conditioned learner is therefore the correct instrument to select the optimal protocol, not a fixed selection or a statically weighted rule.

Safety gates are empirically justified

The separation between learning and reliability in the Materials and Methods section is not only a modeling choice; it is supported by the measurements. In our experimental setup, CoAP shows a sharp loss cliff when the payload approaches and exceeds the supported message-size region around 1900 B (**Fig. 2**). Below this boundary, CoAP remains competitive and can provide low-latency communication for small messages. Once the payload moves beyond the supported range, however, loss increases abruptly and can approach total failure.

This behavior should not be interpreted as a universal fixed limit of the CoAP protocol itself. The observed cliff is implementation- and platform-dependent. CoAP performance near large payloads is affected by the endpoint configuration and library behavior, including message-size limits, resource-body limits, block-wise transfer handling, UDP packet sizing and fragmentation, receive buffers, timeout behavior, and memory available on the edge device. These factors may differ across hardware platforms, operating systems, programming languages, and CoAP libraries.

The CoAP result illustrates why the selector must distinguish between learning which admissible protocol is best and detecting protocols that are no longer operationally safe. When a protocol enters a failure region caused by payload limits or resource constraints, continued exploration does not provide useful performance information. Instead, it mostly produces failed requests and distorted reward observations. A soft reward penalty alone is insufficient in this case: LinUCB may still occasionally re-sample a mechanically failing arm because exploration is part of its decision rule.

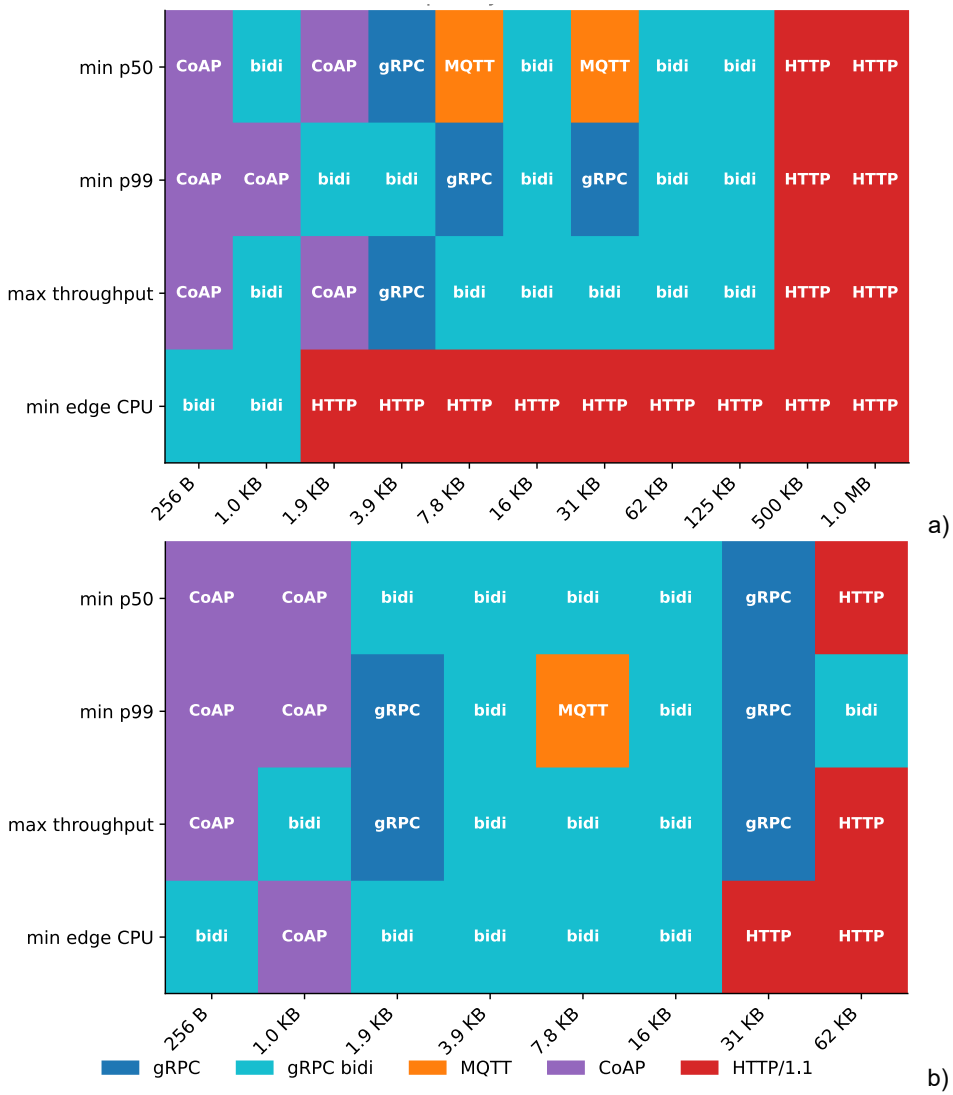

Fig. 1. Best-protocol grid by payload size and optimization objective (p_{50} RTT, p_{99} RTT, throughput, edge CPU): (a) Raspberry Pi 5B; (b) Raspberry Pi Zero 2 W. The winning transport changes across cells, so no single protocol dominates.

LinUCB convergence: a worked example

A clean convergence example is the Raspberry Pi Zero 2 W throughput run at a 256 B payload. Its decision log contains 282 scored UCB decision entries, excluding fallback/event records, with 51 switches; the selected-arm share concentrates on CoAP (≈78.7 %), with MQTT (≈9.6 %) and gRPC-bidi (≈8.9 %) absorbing most of the remainder. The aggregate outcome is p_{50} = 3.548 ms, p_{99} = 18.336 ms, throughput = 198.6 msg/s, and zero effective loss. **Fig. 3** plots the per-arm total UCB index over time. The controller begins in an exploration-dominated phase in which the indices are close and selection alternates, then the exploitation term $\theta^{\mathrm{T}}x$ separates the arms and selection concentrates on the context-conditioned best protocol. This is the exploitation/exploration decomposition of the UCB index made directly observable on real traffic, and it shows that the bandit

leaves its initial exploration phase and locks onto a strong arm for the observed context without external tuning.

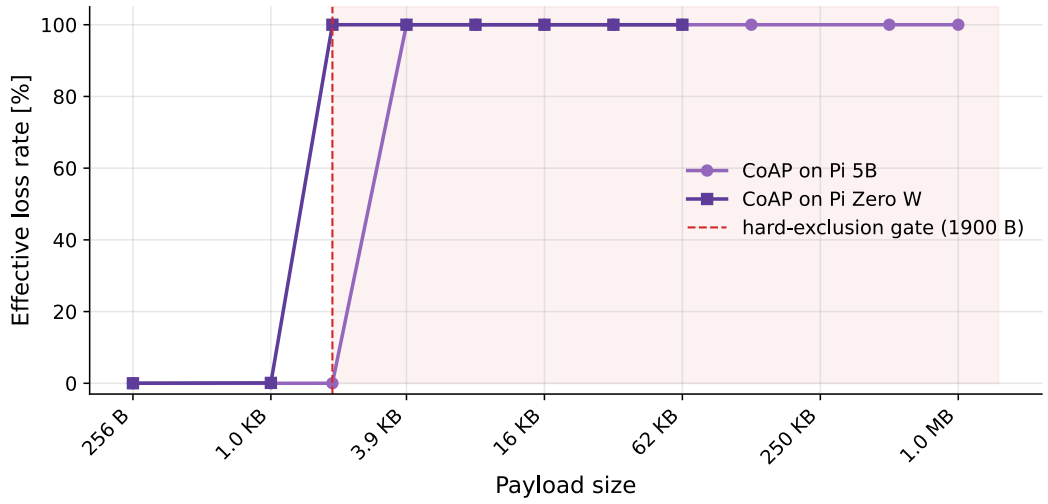

Fig. 2. CoAP message-loss rate versus payload size. The sharp drop just above the 1900 B supported message boundary motivates the hard CoAP payload exclusion (for payloads strictly above 1900 B) in the viability filter, independent of estimated reward.

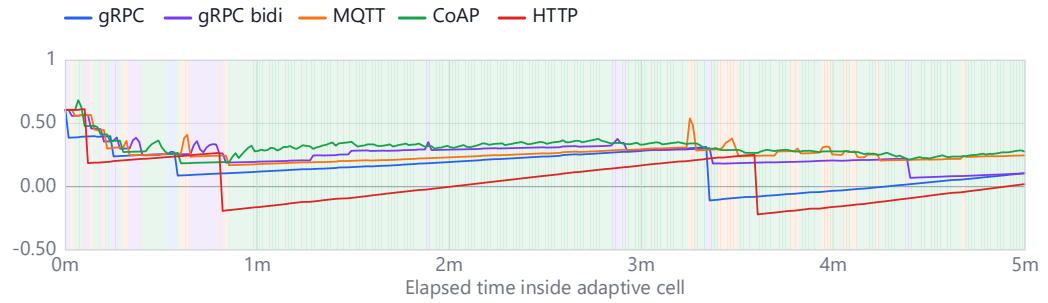

Fig. 3. Per-arm total UCB index over the decision sequence for the Raspberry Pi Zero 2 W 256 B throughput run (282 scored decisions, 51 switches). Each curve is the total index for one protocol arm; the controller begins in an exploration-dominated phase where indices are near-tied and selection alternates, after which the exploitation term ($\theta^{\mathrm{T}}x$) separates the arms and selection concentrates on CoAP (≈78.7% share; MQTT ≈9.6%, gRPC-bidi ≈8.9%). Aggregate outcome: p_{50} = 3.548 ms, p_{99} = 18.336 ms, throughput = 198.6 msg/s, zero effective loss.

Cold-start path dependence

The Pi 5B adaptive sweep illustrates the cold-start behavior of an online learner. At 1900 B, two cold runs followed different trajectories: run 1 concentrated on MQTT (75.3 % share) and missed the isolated-baseline envelope – p_{50} 3.271 ms, a +37.8 % p_{50} gap against the CoAP p_{50} reference (CoAP is still admissible at exactly 1900 B – the hard filter excludes it only strictly above that limit – so it is a legitimate reference here) and a +30.7 % throughput gap against the best logged-candidate reference, gRPC-bidi, while run 2 concentrated on gRPC (90.2 % share) and recovered to near-reference performance (p_{50} 2.397 ms, +1.0 % gap; throughput gap +2.9 %). At 8000 B both repeated runs stayed close

to the isolated-baseline envelope despite different gRPC/gRPC-bidi shares at 128000 B the adaptive runs showed moderate p_{50} gap (≈ +8 %) but a p_{99} that was 8–13 % below the isolated HTTP p_{99} suggesting that looking at median latency alone misses part of the picture, and that the isolated baselines should be read as reference points, not as a head-to-head comparison run at the same time **Fig. 4 (a–d)** shows which protocol was chosen at each decision step for these difficult cases, and **Table 1** puts numbers on how far they were from the baselines. Bad outcome of run-1 is left in deliberately, it is proof that the learner can go wrong early on when it has little data, and leaving it out would make the method look more reliable than it is at startup.

Bayesian prior initialization

The Raspberry Pi Zero 2 W 8000 B comparison gives the clearest evidence that prior initialization changes early adaptive behavior, and it links directly to the warm-start identity (Eq. 21) and (Eq. 22). The no-prior pair converged almost entirely to HTTP (82.8 % and 99.0 % share) and produced higher median and tail RTT; with the prior enabled, the selected-arm distribution shifted toward gRPC/gRPC-bidi. Averaged across the two runs in each condition (n = 2 runs per condition, single host and payload, no confidence intervals – these figures are illustrative of the proof-of-concept), the prior increased

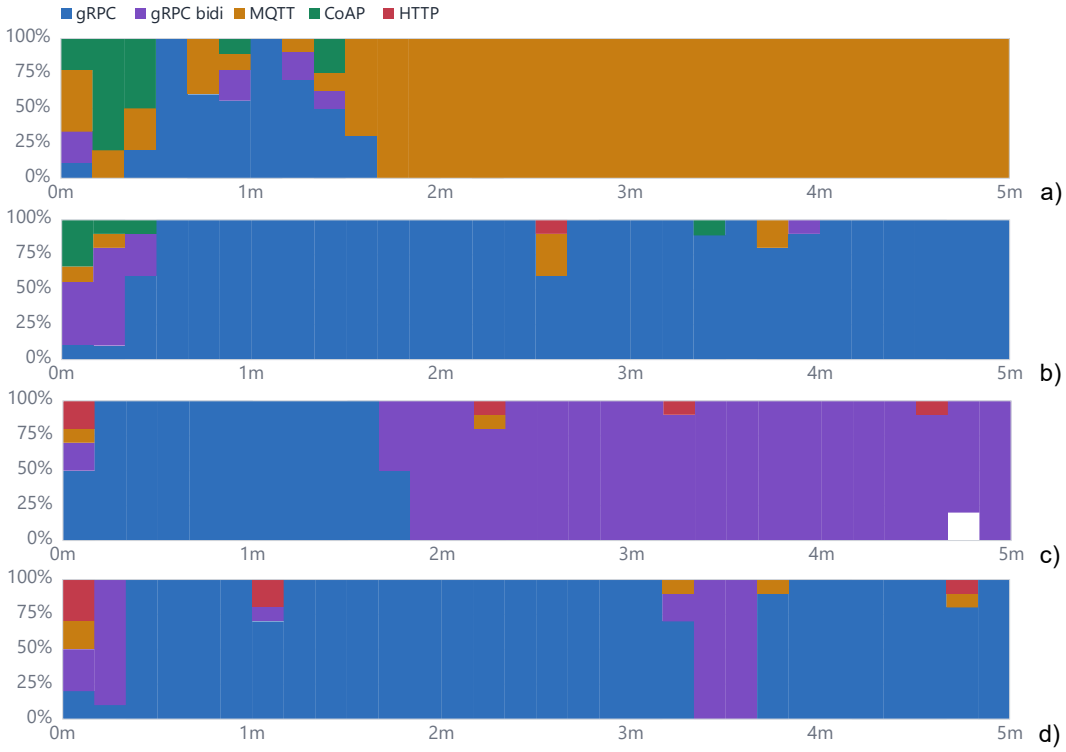

Fig. 4. Selected-arm convergence for the Raspberry Pi 5B hard cases, showing which protocol was chosen at each decision step: (a) 1900 B, cold-start run 1 – selection concentrates on MQTT (≈75.3% share), missing the isolated-baseline envelope (p_{50} 3.271 ms, +37.8% p_{50} gap vs. the CoAP reference); (b) 1900 B, cold-start run 2 – selection concentrates on gRPC (≈90.2% share) and recovers to near-reference performance (p_{50} 2.397 ms, +1.0% gap); (c) 8000 B – selection settles on gRPC-bidi, staying close to the isolated-baseline envelope; (d) 128000 B – selection concentrates on gRPC, with a moderate p_{50} gap (≈+8%) while p_{99} runs 8–13% below the isolated HTTP p_{99}. The (a)/(b) pair shows divergent cold-start trajectories from identical starting conditions.

Table 1. **Selected adaptive runs versus the isolated-protocol baseline reference**

Host	Payload (B)	Run / prior	Dominant arm	p_{50} (ms)	p_{50} gap (%)	Tput gap (%)	p_{99} vs ref. (%)
Pi 5B	1900	run 1, no prior	MQTT 75.3 %	3.271	+37.8	+30.7	+189.5
Pi 5B	1900	run 2, no prior	gRPC 90.2 %	2.397	+1.0	+2.9	+2.2
Pi 5B	8000	run 2, no prior	gRPC-bidi 64.3 %	4.144	+1.4	+1.4	+1.8
Pi 5B	128000	run 2, no prior	gRPC 84.5 %	20.974	+8.3	+5.1	−13.4
Pi Zero 2 W	8000	no prior (mean)	HTTP-dominated	12.401	+14.9..23	+14..20.7	+80..104
Pi Zero 2 W	8000	prior (mean)	gRPC / gRPC-bidi	10.636	−1.3..+5.3	−1.6..+4.6	+27..46

Note: reference = mean of three isolated baseline runs for the same host and payload; positive = worse than reference.

throughput by 19.2 % (70.5 → 84.1 msg/s), decreased p_{50} by 14.2 % (12.40 → 10.64 ms), and decreased p_{99} by 28.9 % (50.43 → 35.87 ms); mean edge CPU rose slightly (8.90 % → 9.26 %), which is reported as an explicit trade-off rather than hidden. **Fig. 5** (per-arm total index) and **Fig. 6** (selected-arm distribution) show that the prior front-loads useful exploitation by initializing each arm from the offline prior, while online evidence still revises the ranking, exactly the graceful-fade behaviour the prior-strength scalar is designed to produce.

Isolated-baseline interpretation and limitations

The results show that no single application protocol is always the best choice. Protocol performance changes with the operating conditions, such as the device, payload size, and measured metric. For this reason, we evaluate the adaptive LinUCB runs against an *isolated baseline envelope*. This envelope is built by taking the best fixed protocol for each host, payload size, and metric from the isolated baseline experiments.

This comparison helps show how close the adaptive selector comes to the best observed fixed protocol result in each case. In many selected runs, the adaptive method performs near this reference envelope. At the same time, the results also show the expected weakness of an online learner at the start of a run: before enough evidence has been collected, early decisions can depend strongly on the first observations.

This baseline should not be interpreted as a direct competitor measured under the same conditions. The fixed-protocol baselines were measured separately and sequentially, while the adaptive selector made decisions inside its own run. Therefore, the baseline envelope is used as a reference for interpretation, not as a true simultaneous counterfactual. For this reason, we do not claim that the adaptive method always outperforms the best fixed protocol.

Five limitations are important in this interpretation.

- *First*, cold-start behavior can lead to poor early choices. The Pi 5B result at 1900 B in run 1 is kept showing that the learning path can depend on the evidence collected at the beginning of the run.

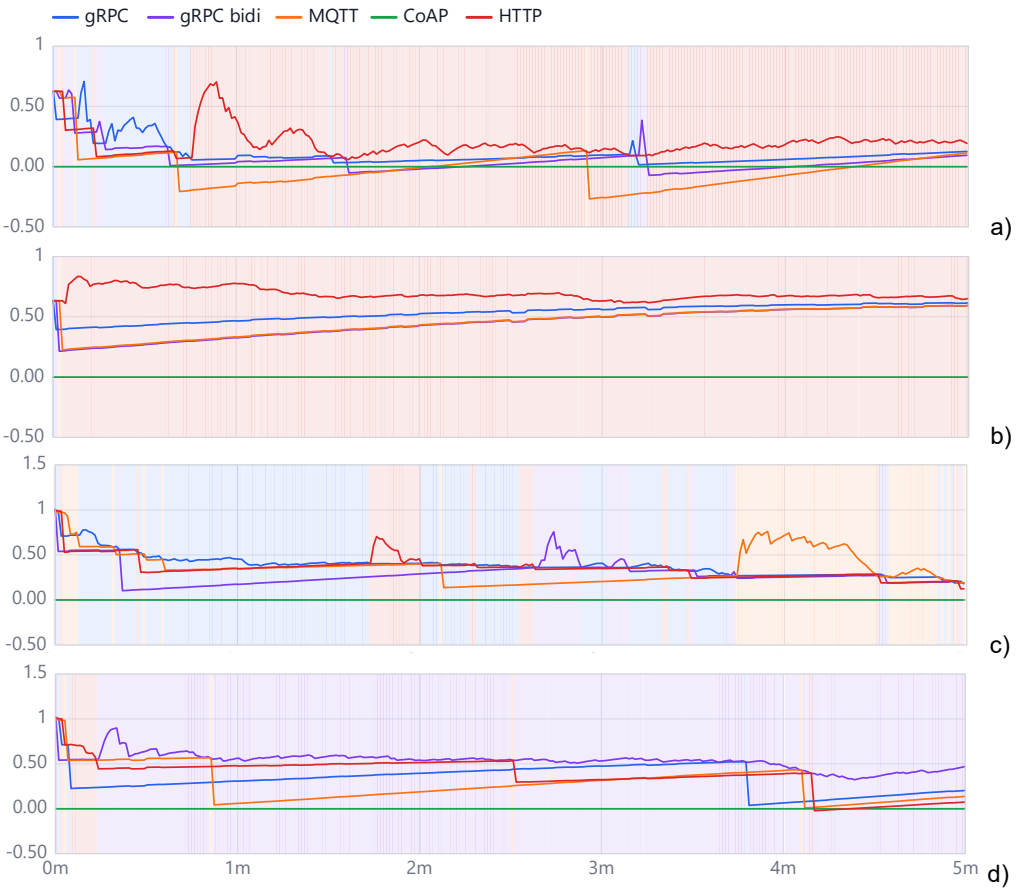

Fig. 5. Per-arm total UCB index over time for the Raspberry Pi Zero 2 W at 8000 B, showing how the Bayesian warm-start reshapes early exploitation: (a) no prior, run 1 – the index trajectories keep HTTP ahead through the cold-start phase, matching the HTTP-dominated selection in Fig. 6-a; (b) no prior, run 2 – the second no-prior run shows the same qualitative HTTP-leading trajectory; (c) with prior, run 1 – the warm-start initializes each arm from the offline prior, front-loading the gRPC/gRPC-bidi indices and shifting the initial policy away from the HTTP-dominated cold trajectory; (d) with prior, run 2 – the second prior-enabled run shows the same qualitative front-loaded gRPC/gRPC-bidi indices, with the warm-start redirecting early exploitation in this run as well.

- *Second*, a Bayesian prior can improve some performance metrics while worsening others. In the Raspberry Pi Zero 2 W 8000 B case, the prior improves throughput, median latency (p_{50}), and tail latency (p_{99}), but slightly increases mean CPU usage. Whether this trade-off is acceptable depends on the system objective. For a CPU-constrained edge device, the operator may prefer a different balance.
- *Third*, the CoAP loss cliff should not be read as a failure of the bandit learner. Instead, it shows that some protocols can become mechanically unsuitable in certain regimes. Such cases must be handled by a hard viability filter that removes unsafe or non-viable actions, rather than relying only on a softer reward penalty.
- *Fourth*, the reported warm-start gains are illustrative rather than statistically characterized. The throughput, p_{50}, and p_{99} improvements above are averaged over only two runs in a single host and payload condition, without confidence intervals or multi-seed replication. A systematic multi-seed study with variance estimates is deferred to the companion paper.

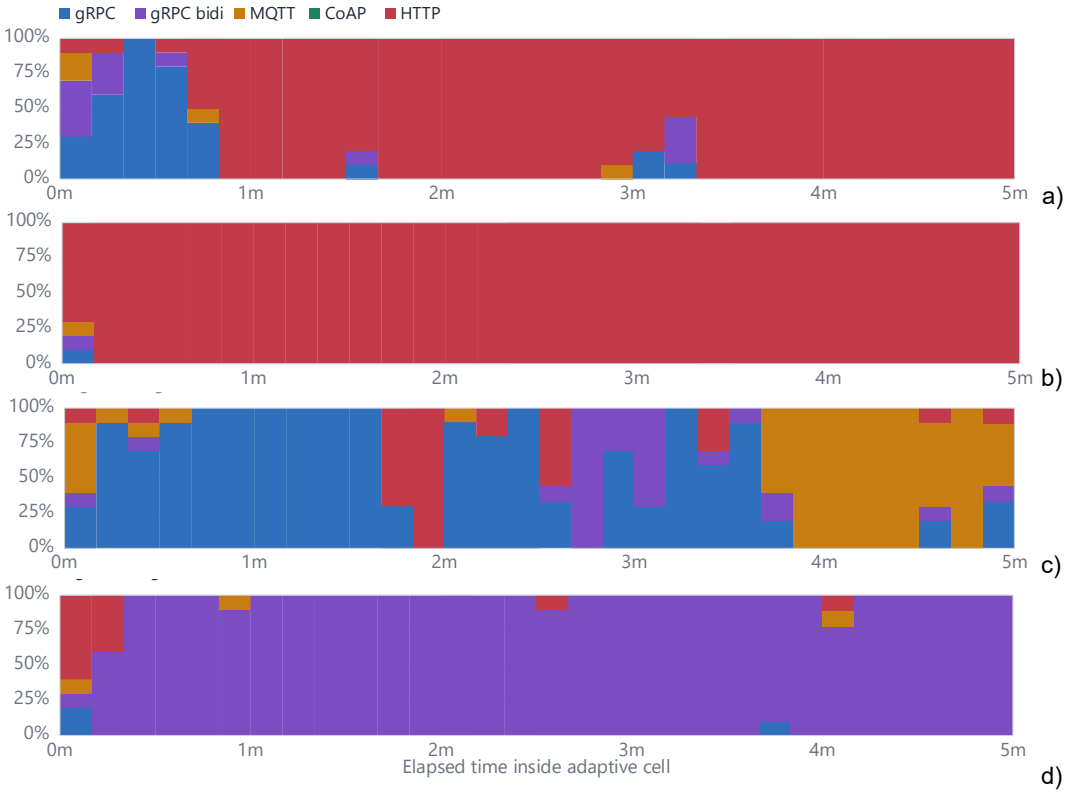

Fig. 6. Selected-arm distribution over time for the Raspberry Pi Zero 2 W at 8000 B, no prior versus Bayesian warm-start: (a) no prior, run 1 – selection collapses almost entirely onto HTTP, reflecting the cold-start trajectory before sufficient online evidence is gathered, with gRPC and gRPC-bidi receiving few selections and yielding higher median and tail RTT; (b) no prior, run 2 – the second no-prior run shows the same qualitative HTTP-dominated pattern; (c) with prior, run 1 – the warm-start shifts the distribution away from HTTP toward gRPC/gRPC-bidi as each arm is initialized from the offline prior, front-loading exploitation while online evidence still revises the ranking, lowering median and tail RTT and raising throughput; (d) with prior, run 2 – the second prior-enabled run shows the same qualitative shift toward gRPC/gRPC-bidi, consistent with the throughput and RTT gains reported across the two-run average.

- *Fifth*, on wired-link deployments (such as tested Raspberry Pi 5B connected via Gigabit Ethernet), the context collapses to [payloadNorm, bias], as noted in the Methods. The worked examples reported here are therefore not an evaluation of the full four-dimensional context under wired conditions: the Pi 5B results reflect only payload-driven arm discrimination, not link-quality discrimination. A wired-aware context extension is planned for the companion paper.

Overall, these examples make the main parts of the approach visible on real hardware: the balance between exploration and exploitation in the UCB score, the effect of Bayesian warm-starting, and the separation between learning and safety constraints. They also define the baseline interpretation used for the broader evaluation in the companion study, including payload and rate sweeps, offline replay experiments, reward-shaping comparisons, and exploration-parameter ablations.

CONCLUSION

We have established the theoretical foundation for an adaptive protocol selection layer for cloud–edge data delivery. Transport choice at adaptive decision ticks is cast as a contextual multi-armed bandit and solved with LinUCB, whose decision index cleanly

separates exploitation (the ridge estimate $\theta^{\mathrm{T}}x$) from exploration (the uncertainty width $\alpha \cdot \sqrt{x^T A^{-1} x}$. The context is a compact four-dimensional vector, the reward captures tail latency, packet loss, and throughput in two bounded formulations an *absolute mode* that scores each protocol against fixed thresholds, and a *relative mode* that scores it against the current field. Non-stationarity is handled by a geometric decay that gradually discounts old observations. A Bayesian warm start lets the selector begin from an offline prior and shift toward live data at a tunable rate. Unsafe protocols are blocked by a hard filter before the learner acts, meaning the safety is not mixed into the reward signal. The implementation supports five concrete transports. Each formula in the paper has a direct counterpart in the code, so the results are fully reproducible given the same configuration and candidate protocols. We validated the formulation on a heterogeneous Raspberry Pi 5B / Raspberry Pi Zero 2 W testbed, showing regime dependence in the baselines, the exploitation/exploration split in the decision logs, and cold-start path dependence in the repeated runs. The companion paper will report the full payload/rate sweep, the replay parameter study, and the reward-shaping and exploration-coefficient sensitivity analyses and will extend the evaluation to dynamic conditions including mid-run wireless network changes and varying payload sizes within a single run. It will also extend the context vector for wired-link deployments – replacing the two Wi-Fi features with wired-link analogues (interface speed, instantaneous goodput, link-saturation indicator) already exposed by the edge reporter. This paper's formulation serves as the fixed methodological baseline for that study.

ACKNOWLEDGMENTS AND FUNDING SOURCES

The authors received no financial support for the research, authorship, and/or publication of this article.

COMPLIANCE WITH ETHICAL STANDARDS

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

AUTHOR CONTRIBUTIONS

Conceptualization, [R.N., I.B.]; methodology, [R.N.]; software, [R.N.]; formal analysis, [R.N.]; investigation, [R.N.]; writing – original draft, [R.N.]; writing – review and editing, [I.B.]; supervision, [I.B.].

All authors have read and agreed to the published version of the manuscript.

REFERENCES

[1] Nazarevych, R., & Bolesta, I. (2025). Software of Internet-accessible semiconductor laboratory. *Information Systems and Networks, 17*, 146–159 (In Ukrainian). https://doi.org/10.23939/sisn2025.17.146

[2] gRPC Authors. (2024). *gRPC: A high-performance, open-source universal RPC framework – Documentation*. Cloud Native Computing Foundation. Retrieved May 17, 2026. https://grpc.io/docs/

[3] OASIS. (2019). *MQTT version 5.0* (OASIS Standard). Organization for the Advancement of Structured Information Standards. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

[4] Shelby, Z., Hartke, K., & Bormann, C. (2014). *The Constrained Application Protocol (CoAP)* (RFC 7252). Internet Engineering Task Force. https://doi.org/10.17487/RFC7252
[5] Iyengar, J., & Thomson, M. (2021). *QUIC: A UDP-based multiplexed and secure transport* (RFC 9000). Internet Engineering Task Force. https://doi.org/10.17487/RFC9000
[6] Bishop, M. (2022). *HTTP/3* (RFC 9114). Internet Engineering Task Force. https://doi.org/10.17487/RFC9114
[7] Naik, N. (2017). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *2017 IEEE International Systems Engineering Symposium (ISSE)* (pp. 1–7). IEEE. https://doi.org/10.1109/SysEng.2017.8088251
[8] Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning, 47*(2–3), 235–256. https://doi.org/10.1023/A:1013689704352
[9] Lai, T. L., & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics, 6*(1), 4–22. https://doi.org/10.1016/0196-8858(85)90002-8
[10] Lattimore, T., & Szepesvári, C. (2020). *Bandit algorithms*. Cambridge University Press. https://doi.org/10.1017/9781108571401
[11] Bouneffouf, D., Rish, I., & Aggarwal, C. (2020). Survey on applications of multi-armed and contextual bandits. In *2020 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1–8). IEEE. https://doi.org/10.1109/CEC48606.2020.9185782
[12] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press. http://incompleteideas.net/book/the-book-2nd.html
[13] Slivkins, A. (2019). Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning, 12*(1–2), 1–286. https://doi.org/10.1561/2200000068
[14] Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)* (pp. 661–670). ACM. https://doi.org/10.1145/1772690.1772758
[15] Chu, W., Li, L., Reyzin, L., & Schapire, R. E. (2011). Contextual bandits with linear payoff functions. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)* (Vol. 15, pp. 208–214). PMLR. https://proceedings.mlr.press/v15/chu11a.html
[16] Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics, 12*(1), 55–67. https://doi.org/10.1080/00401706.1970.10488634
[17] Golub, G. H., & Van Loan, C. F. (2013). *Matrix computations* (4th ed.). Johns Hopkins University Press. https://doi.org/10.56021/9781421407944
[18] Abbasi-Yadkori, Y., Pál, D., & Szepesvári, C. (2011). Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems (NeurIPS)* (Vol. 24, pp. 2312–2320). Curran Associates. https://proceedings.neurips.cc/paper/2011/hash/e1d5be1c7f2f456670de3d53c7b54f4a-Abstract.html

[19] Garivier, A., & Moulines, E. (2011). On upper-confidence bound policies for switching bandit problems. In *Algorithmic Learning Theory (ALT 2011), LNCS* (Vol. 6925, pp. 174–188). Springer. https://doi.org/10.1007/978-3-642-24412-4_16

[20] Shivaswamy, P., & Joachims, T. (2012). Multi-armed bandit problems with history. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)* (Vol. 22, pp. 1046–1054). PMLR. https://proceedings.mlr.press/v22/shivaswamy12.html

[21] Kazerouni, A., Ghavamzadeh, M., Abbasi-Yadkori, Y., & Van Roy, B. (2017). Conservative contextual linear bandits. In *Advances in Neural Information Processing Systems (NeurIPS)* (Vol. 30, pp. 3910–3919). Curran Associates. https://proceedings.neurips.cc/paper/2017/hash/bdc4626aa1d1df8e14d80d345b2a442d-Abstract.html

МЕТОД АДАПТИВНОГО ВИБОРУ ПРОТОКОЛІВ ДЛЯ КОНТЕКСТНО-ЗАЛЕЖНОЇ ПЕРЕДАЧІ ДАНИХ У ХМАРНО-ПЕРИФЕРІЙНИХ СИСТЕМАХ НА ОСНОВІ КОНТЕКСТНОГО БАНДИТА З БАЄСОВОЮ АПРІОРНОЮ ІНІЦІАЛІЗАЦІЄЮ

Роман Назаревич, Іван Болеста
Кафедра радіофізики та комп'ютерних технологій,
Львівський національний університет імені Івана Франка,
вул. ген. Тарнавського, 107, Львів, 79017, Україна

АНОТАЦІЯ

Вступ. Продуктивність протоколів у хмарно-периферійних системах залежить від розміру корисного навантаження, інтенсивності передавання, типу каналу та якості сигналу. Тому протокол, оптимальний за затримкою, втратами або витратами процесорного часу, змінюється разом з умовами роботи, а ранжування за хвостовою затримкою може не збігатися з ранжуванням за медіанною. Статичні механізми й фіксовані евристики не забезпечують належної адаптації, а більшість наявних «адаптивних» рішень є детермінованими оцінювачами, а не системами онлайн-навчання. У статті обґрунтовано адаптивний метод, що формулює вибір протоколу як задачу онлайн-прийняття рішень за часткового зворотного зв'язку. Метод продемонстровано на концептуальних прикладах; повну експериментальну оцінку буде наведено в супровідній статті.

Матеріали та методи. Вибір протоколу сформульовано як задачу контекстного багаторукого бандита для п'яти транспортів і розв'язано алгоритмом LinUCB. Кожен протокол моделюється окремою гребеневою регресією з верхньою довірчою межею, параметри якої обчислюються методом Ґауса. Контекст охоплює логарифмічно нормалізований розмір корисного навантаження, нормалізовані рівень Wi-Fi-сигналу та швидкість передавання, а також вільний член. Винагорода враховує медіанну й хвостову затримку, корисну пропускну здатність і втрати. Нестаціонарність моделюється геометричним згасанням, а баєсова апріорна ініціалізація задає початкові параметри регресії через матрицю точності. Оскільки згасання застосовується покроково, апріорне знання може послаблюватися ще до першої оцінки. Формули узгоджені з програмною реалізацією.

Результати та обговорення. Запропоновану формалізацію для п’яти транспортних протоколів продемонстровано на ізольованих експериментальних прикладах із використанням одноплатних комп’ютерів Raspberry Pi 5B та Pi Zero 2 W. У журналах адаптивного вибору розглядали п’ять протоколів передавання даних: gRPC, gRPC-bidi, MQTT, CoAP і HTTP. Базові експерименти підтвердили, що оптимальний протокол залежить від розміру корисного навантаження, апаратної платформи та цільової функції. Аналіз журналів показав, що в стабільних режимах LinUCB зосереджує вибір на протоколі, найефективнішому для поточного контексту. Водночас приклад невдалого холодного старту продемонстрував чутливість алгоритму до ранніх спостережень. Баєсова апріорна ініціалізація зменшила цей ефект і поліпшила показники затримки та пропускної здатності на початковому етапі.

Висновки. У статті запропоновано відтворюване формулювання адаптивного вибору протоколу для передавання даних між хмарними та периферійними вузлами на основі контекстного багаторукого бандита з баєсовою апріорною ініціалізацією. Визначення контексту, функції винагороди та стратегії дослідження забезпечує незалежне відтворення методу й формує основу для дослідження в супровідній статті.

Ключові слова: контекстний бандит, LinUCB, адаптивний вибір протоколу, хмарно-периферійні системи, баєсовий апріорний розподіл, компроміс дослідження–експлуатація.

Received / Одержано	Revised / Доопрацьовано	Accepted / Прийнято	Published / Опубліковано
26 May, 2026	25 June, 2026	26 June, 2026	29 June, 2026