# Toward Forecast-Aware LLM Agents for Network Management

Jaehyung Choi*, Taeil Jung*, Kyung Sook Kim†, Jeehyeon Na†, Een-Kee Hong*
*Department of Electronics and Information Convergence Engineering, Kyung Hee University, Yongin, Republic of Korea
Email: {wogud1221,wjdxodlf012345,ekhong}@khu.ac.kr
†Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea
Email: {newday,jhna}@etri.re.kr

***Abstract*—Large language models (LLMs) are increasingly used as network management agents that reason over telemetry and assist decision-making. If such agents are to enable prediction-driven control, they must obtain short-term forecasts of key performance indicators (KPIs) under tight operational constraints, notably accuracy and low end-to-end latency. A naïve approach is to let the LLM forecast numeric sequences directly (via prompting or fine-tuning), but this can be unstable for long horizons and often requires task-specific tuning. In this paper, we move toward *forecast-aware* LLM agents by adopting a tool-augmented design: the LLM orchestrates forecasting requests (when to forecast, which horizon, and how to use the output), while forecasting itself is delegated to an external model. We study this design choice through a cellular-traffic case study, comparing (i) LLM-only forecasting (Mistral 7B), (ii) an LLM calling a conventional LSTM forecaster trained per cell, and (iii) an LLM calling an off-the-shelf, pre-trained time-series foundation model, Chronos-2. Results show that LLM-only forecasts drift over a multi-day horizon, and the LSTM tool improves accuracy at a higher end-to-end cost due to training and invocation overhead. In contrast, Chronos-2 delivers the best accuracy–latency trade-off without per-cell retraining, supporting the practicality of tool-based forecasting back-ends for forecast-aware LLM agents in network management.**

***Keywords*—LLM agent, Network Management, Forecasting, Tool Augmentation, Time-Series Foundation Model, Chronos-2**

## I. Introduction

Next-generation mobile networks operate under highly dynamic and heterogeneous conditions, where traffic demand, interference, and quality-of-experience (QoE) can fluctuate over short time scales. In such environments, *prediction* becomes a practical requirement for network management rather than an optional feature: traffic and load balancing, anomaly detection, proactive resource reservation, energy-aware cell control, and service-level agreement (SLA) assurance all benefit from predicting the near-future evolution of key performance indicators (KPIs) such as throughput, latency, error rates, and user demand. While our case study focuses on cell-level traffic, the same need applies broadly to KPI-driven automation in 5G/6G systems [1], [2].

LLMs are increasingly explored as *network-management agents*. To support prediction-driven control, such agents must be able to obtain short-term KPI forecasts and incorporate them into actionable decisions. Importantly, forecasts used in operational settings are constrained by more than accuracy alone: an agent must produce forecasts within tight runtime budgets (end-to-end latency), and the forecasting mechanism should remain usable across many cells, horizons, and KPIs without expensive per-case retraining or prompt redesign. This operational perspective aligns with ongoing efforts toward practical, deployable RAN control platforms and data-driven automation [3].

A naïve approach is to ask the LLM itself to output future KPI values, either by fine-tuning the LLM on numeric sequences or by prompting it to extrapolate raw values. However, *LLM-only* forecasting can be brittle in practice: general-purpose LLMs are not optimized for numerical time-series modeling, and long-horizon outputs may drift or become unstable without careful, task-specific tuning. Moreover, repeatedly generating long numeric sequences through an LLM can be computationally costly, which conflicts with the low-latency requirements of network control loops. These concerns are consistent with broader observations on reliability and robustness challenges in LLM-empowered communication systems [4].

An alternative is to treat the LLM primarily as an *orchestrator* and delegate numerical forecasting to specialized models accessed through tool interfaces. This tool-augmented paradigm is well established in the broader LLM ecosystem, but its implications for network KPI forecasting merit careful study because network agents face strict accuracy–latency trade-offs and operational constraints. Meanwhile, the time-series community has produced *pre-trained* forecasting models designed for multi-series and multi-horizon prediction in a plug-and-play manner. Chronos-2 is one representative example: a probabilistic transformer-based forecaster trained on large collections of time series and intended to be used without task-specific retraining. In this paper, we do *not* introduce Chronos-2 as a new model; instead, we use it as an off-the-shelf forecasting back-end to examine how LLM-based network agents should obtain forecasts in practice.

Specifically, we conduct a cellular-traffic case study using a public dataset and compare three strategies that differ only in how forecasts are produced: (i) **LLM-only** forecasting using a general-purpose model (Mistral 7B), (ii) an LLM calling a **conventional LSTM** forecaster trained on the target series, and (iii) an LLM calling **Chronos-2** as a pre-trained forecasting back-end. For each strategy, we report both fore-

casting error (MAE) and end-to-end prediction time, reflecting the agent-centric view that forecasts must be accurate *and* available quickly enough for operational use. Our results show that LLM-only forecasting is prone to long-horizon drift, and that while an LSTM-based tool can improve accuracy, it introduces additional overhead due to task-specific training and invocation. In contrast, the pre-trained Chronos-2 back-end provides a favorable accuracy–latency trade-off without per-cell retraining, suggesting that pre-trained forecasting tools are a practical default choice for building forecast-aware LLM agents in network management.

Based on this case study, the contributions of this work can be summarized as follows:

- We motivate an agent-centric view of KPI forecasting for *LLM-based* network management, emphasizing operational requirements beyond accuracy, including end-to-end latency and reusability across settings.
- We adopt a tool-augmented design toward forecast-aware LLM agents, where the LLM orchestrates forecasting requests while numerical prediction is delegated to external forecasters.
- We provide an empirical comparison of three forecasting strategies (LLM-only, LLM+LSTM, and LLM+Chronos-2) on cellular traffic traces, reporting both MAE and end-to-end prediction time.
- We discuss practical implications for deploying forecast-aware LLM agents, highlighting when tool-based forecasting back-ends are preferable to relying on LLM-only prediction.

## II. Related Work

Forecasting has long been recognized as an enabling capability for network operation and control, where predicted traffic or KPI trajectories are consumed by downstream decision modules (e.g., resource provisioning, energy saving, and sleep control). Recent work on cellular traffic forecasting has explicitly connected prediction quality to operational objectives, demonstrating that forecast accuracy can be a practical prerequisite for control policies such as base-station sleep scheduling [5]. Such studies highlight that forecasting in networks is rarely an end in itself; rather, it is a supporting function that must be sufficiently accurate and available within a time budget compatible with operational decision loops.

LLMs have also begun to be explored as agents for network management, aiming to translate high-level intents into actionable configurations and procedures while improving reliability through structured execution and guardrails [6]. MeshAgent is a representative example that targets network-management tasks that can be framed as graph manipulation, and it improves reliability by constraining and validating LLM-generated actions [7]. Importantly, MeshAgent also delineates a clear limitation: certain operational tasks, such as flow-level performance monitoring that requires statistical analysis over time-series measurements (e.g., latency, jitter, and loss), are better handled by signal-processing or time-series models than by graph-oriented reasoning [7]. This observation supports the view that a practical network LLM agent will rely on specialized external modules for time-series inference, including KPI forecasting.

Tool-augmented LLM agents provide a general mechanism to realize such specialization. Frameworks such as ReAct formalize an agent pattern in which an LLM interleaves natural-language reasoning with actions that query external tools and environments [8]. In this paradigm, the LLM is not required to internally master every numerical or domain-specific skill; instead, it orchestrates when to invoke specialized functions and how to interpret their outputs. This provides a principled basis for positioning forecasting as a callable capability within an LLM-driven network agent. Related efforts also study how to structure and optimize tool-augmented LLM pipelines for practical use [9].

Recent progress on pre-trained forecasting models further strengthens the case for tool-based prediction. While classic neural forecasters (e.g., LSTMs) can be effective, they typically require task- and series-specific training and tuning, which introduces non-trivial overhead when deployed as an on-demand tool inside an agent loop. In contrast, forecasting foundation models such as Chronos and Chronos-2 are designed to be used in a plug-and-play manner across diverse series and horizons, providing probabilistic forecasts without per-series retraining [10], [11]. This property aligns closely with the operational requirements of network LLM agents, where forecasts must be obtained quickly and repeatedly across many cells and KPIs. Motivated by these trends, this paper adopts an agent-centric view of forecasting and examines practical strategies for obtaining forecasts under operational accuracy–latency constraints.

## III. Experimental Setup

To assess how an LLM-based network agent should obtain network data forecasts, we conduct an *agent-centric* case study in which forecasting is treated as a callable capability used to support downstream management decisions. In this setting, the forecasting component is not evaluated solely by prediction accuracy; rather, it must also satisfy operational constraints such as responsiveness, since an agent may need to request forecasts repeatedly across many cells and time horizons. Accordingly, our evaluation jointly considers (i) forecasting error over the requested horizon and (ii) the end-to-end time perceived by the agent to obtain the forecast, enabling an accuracy–latency interpretation that is aligned with practical agent operation in network management.

### A. Dataset and KPI Construction

We use the public Milan cellular dataset, which reports cell-level usage statistics in ten–minute intervals from 1 December 2013 to 1 January 2014. For each target cell, we aggregate the raw counters (SMS, calls, and Internet usage) into a single scalar value per time slot, producing a univariate time series that represents the overall traffic intensity of that cell. This simple KPI construction is intended to reflect a practical scenario in which an agent consumes a compact, single-stream

indicator of cell activity (rather than multiple heterogeneous counters) when making rapid management decisions. Although our case study uses traffic, the same experimental structure applies to generic network KPI prediction where an agent must forecast the near-future evolution of a measurable quantity from its history [12].

### B. Forecasting Task and Horizon

We split the time series into a *history window* and a *prediction horizon*. The samples from `2013-12-01` to `2013-12-26` are used as the history, and the final six days (`2013-12-27` to `2014-01-01`) form the horizon to be forecast. Given the observed sequence up to the end of 26 December, the task is to produce a point forecast for every time step in the six–day horizon. We adopt a multi-day horizon to expose differences that are relevant to agent operation: when forecasts are required beyond very short look-ahead steps, direct numeric extrapolation may drift and trained-per-series predictors may incur additional overhead, whereas pre-trained forecasting back-ends are designed to remain usable across horizons without re-training. In this sense, the chosen horizon provides an informative stress case for comparing forecasting strategies under agent-centric constraints.

### C. Agent Configurations

The underlying language model in all experiments is Mistral 7B in FP16 precision. We compare three configurations that differ only in how the forecasts are obtained:

1) **LLM-only**: the LLM receives the past KPI sequence encoded as text and is prompted to output the future values directly, without using any external forecasting model. This configuration represents the most direct approach in which the agent relies on the LLM as a stand-alone numerical forecaster. In our study, it serves as a sanity-check baseline for understanding how a general-purpose LLM behaves when asked to extrapolate a multi-day numeric trajectory and whether it preserves basic properties such as trend and daily periodicity.
2) **LLM + LSTM tool**: the LLM is equipped with a tool that wraps a conventional LSTM forecaster. The LSTM is trained on the history window for the target cell, and at inference time the LLM calls this tool to obtain the predictions for the six–day horizon. This configuration represents a trained-per-series forecasting back-end that is commonly adopted in cellular traffic prediction. From an agent perspective, it provides a useful contrast to pre-trained models because it requires an explicit training step and model invocation before forecasts can be returned to the agent.
3) **LLM + Chronos-2 tool**: the LLM is equipped with a tool that wraps Chronos-2, a pre-trained probabilistic time-series model. Chronos-2 receives the same history window as input but does not require task-specific retraining for the target cell; the LLM simply uses this tool to perform the forecasting step. This configuration represents the tool-augmented design in which forecasting is delegated to an off-the-shelf, pre-trained back-end that can be called in a plug-and-play manner, matching the operational expectation that an agent should obtain forecasts quickly without repeated per-target training.

### D. Evaluation Criteria

For each configuration, we evaluate the forecasts using three criteria from the perspective of the LLM agent. First, we quantify forecasting error over the horizon using the mean absolute error (MAE),

$$\mathrm{MAE} = \frac{1}{T}\sum_{t=1}^{T}\left|y_t - \hat{y}_t\right|, \tag{1}$$

where $y_t$ and $\hat{y}_t$ denote the actual and predicted values at time $t$, and $T$ is the number of predicted samples. Second, we measure the end-to-end prediction time, defined as the wall-clock time from the moment the agent requests a forecast until all horizon values are available to the agent. This end-to-end metric is included because, in operational settings, an accurate forecast that arrives too late may be less useful than a slightly less accurate forecast that can be acted upon immediately. Third, we provide a qualitative visual comparison of the predicted and actual trajectories using time-series plots, which helps interpret whether each method preserves coarse properties such as level, trend, and daily structure. Together, these criteria reflect the practical objective of building *forecast-aware* LLM agents that can obtain forecasts with an appropriate balance between accuracy and responsiveness.

## IV. Results

We evaluate the three forecasting strategies from an agent-centric perspective by jointly considering forecasting error and end-to-end prediction time. We first present a qualitative comparison of forecast trajectories and then summarize the quantitative accuracy–latency trade-off.

### A. Accuracy and qualitative behavior

We report qualitative results using a representative cell whose forecasting error is close to the median across the evaluated cells. This selection is intended to reflect typical behavior under each configuration, rather than highlighting exceptionally good or poor cases.

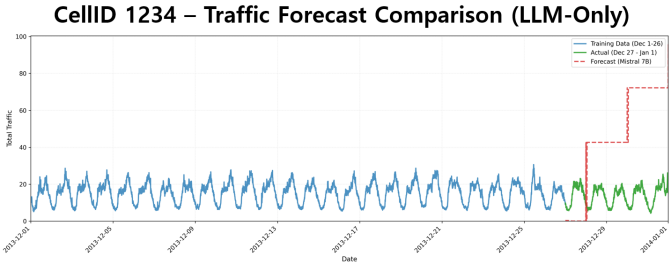


Fig. 1. Qualitative forecasting behavior for a representative cell under the LLM-only configuration.

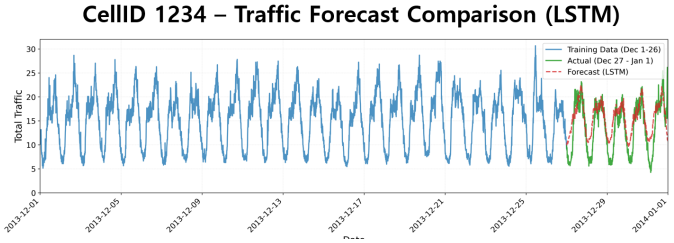


Fig. 2. Qualitative forecasting behavior for the same cell under the LLM + LSTM tool configuration.

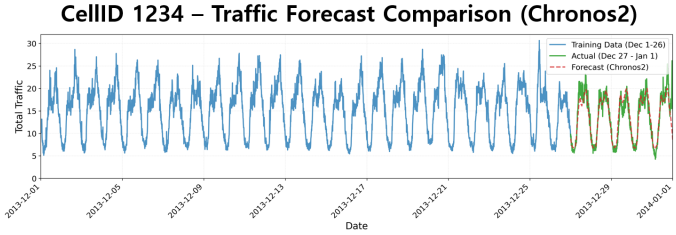


Fig. 3. Qualitative forecasting behavior for the same cell under the LLM + Chronos-2 tool configuration.

| Configuration | MAE | Prediction time [s] |
|---|---|---|
| LLM-only | – | 14.92 |
| LLM + LSTM tool | 2.54 | 48.24 |
| LLM + Chronos2 tool | 1.48 | 12.58 |

Fig. 4. Forecasting performance from an agent-centric perspective. Lower is better for both MAE and end-to-end prediction time.

In the *LLM-only* configuration, the model is prompted to extrapolate the KPI sequence without using any dedicated forecasting module. As shown in Fig. 1, the predicted trajectory exhibits noticeable drift over the multi-day horizon and does not preserve the daily periodic structure observed in the historical data, suggesting that direct numeric extrapolation with a general-purpose LLM can be unstable for multi-day KPI forecasting.

Equipping the agent with an *LSTM tool* improves forecast fidelity. Fig. 2 shows that the predicted series follows the daily ups and downs of the actual traffic more closely than the LLM-only output, resulting in an MAE of **2.54** over the six-day horizon (Fig. 4). This indicates that a trained neural forecaster can capture the dominant temporal pattern when used as an external forecasting back-end in this case study.

The *LLM + Chronos-2 tool* configuration achieves the lowest MAE among the three strategies. As shown in Fig. 3, Chronos-2 closely matches both the level and daily structure of the target series, yielding an MAE of **1.48** without per-cell retraining (Fig. 4). From an agent perspective, this result is consistent with the use of pre-trained forecasting back-ends for producing stable multi-day KPI predictions.

### *B. End-to-end prediction time and accuracy–latency trade-off*

Beyond accuracy, a forecast-aware agent must obtain predictions within a runtime budget compatible with operational decision loops. All timing results were measured on a single machine under identical runtime settings to ensure a fair comparison across configurations. Fig. 4 shows that the LSTM tool incurs the highest end-to-end prediction time (**48.24 s**), reflecting the overhead of using a trained-per-series forecasting back-end in an on-demand tool-calling workflow. In contrast, Chronos-2 yields the fastest end-to-end prediction time (**12.58 s**), which is even shorter than the **14.92 s** observed for LLM-only generation. This suggests that delegating forecasting to a specialized pre-trained model can improve numerical accuracy while also reducing the agent-perceived time to obtain forecasts.

Overall, the comparison indicates that the three strategies exhibit distinct accuracy–latency profiles in our setting, with the pre-trained Chronos-2 back-end offering a particularly favorable balance.

## V. Conclusion

This paper examined how an LLM-based network management agent should obtain short-term KPI forecasts under practical operational constraints. Rather than relying on the LLM as a stand-alone numerical forecaster, we evaluated a tool-augmented design in which the LLM orchestrates forecasting requests while numerical prediction is delegated to an external model. Using a cellular-traffic case study, we compared three strategies: LLM-only forecasting, an LLM calling a trained-per-cell LSTM forecaster, and an LLM calling a pre-trained forecasting foundation model (Chronos-2). Note that our goal is not to propose a new forecasting model, but to clarify how an LLM agent should obtain forecasts in practice under accuracy–latency constraints.

The results highlight a clear agent-centric trade-off. LLM-only forecasting exhibited noticeable drift over a multi-day horizon, indicating limited reliability for direct numeric extrapolation. The LSTM tool improved accuracy (MAE 2.54) but incurred the highest end-to-end prediction time (48.24 s), reflecting the overhead of using a trained-per-series back-end in an on-demand tool-calling workflow. In contrast, Chronos-2 achieved the lowest MAE (1.48) while also yielding the fastest end-to-end prediction time (12.58 s), demonstrating that a plug-and-play, pre-trained forecasting back-end can deliver a favorable accuracy–latency balance for forecast-aware LLM agents without per-cell retraining. This latency advantage is consistent with the fact that tool-based forecasting avoids long sequence generation inside the LLM and returns horizon values through a single external inference step.

These findings support a practical design principle for prediction-driven network automation: LLMs are most effective when used as orchestration layers that invoke specialized forecasting tools, and pre-trained forecasting models are a strong default choice when forecasts must be obtained quickly and repeatedly across many targets. Future work will extend this evaluation to additional KPIs and horizons, assess robustness across diverse traffic regimes, and integrate forecast-aware agents into closed-loop network control pipelines where prediction latency and decision latency are jointly constrained.

An important next step is to validate whether the same accuracy–latency trend holds across a larger set of cells and KPIs (e.g., throughput or latency) and under different horizon lengths.

## VI. Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2024-00395824, Development of Cloud virtualized RAN (vRAN) system supporting upper-midband, 50) and partly supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. RS-2024-00397520, Development a virtualized compact 5G all-in-one system for vertical services in 5G networks)

## References

[1] W. Jiang, "Cellular traffic prediction with machine learning: A survey," *Expert Systems with Applications*, vol. 201, p. 117163, 2022.

[2] X. Wang, Z. Wang, K. Yang, Z. Song, C. Bian, J. Feng, and C. Deng, "A survey on deep learning for cellular traffic prediction," *Intelligent Computing*, vol. 3, no. 2, article 0054, 2024.

[3] M. V. Ngo, N. B. L. Tran, H. M. Yoo, Y. H. Pua, T. L. Le, X. L. Liang, B. Chen, E. K. Hong, and T. Q. S. Quek, "RAN Intelligent Controller (RIC): From open-source implementation to real-world validation," *ICT Express*, vol. 10, no. 3, pp. 680–691, 2024.

[4] T. Liu *et al.*, "Hallucination-aware optimization for large language model-empowered communications," *IEEE Transactions on Wireless Communications*, early access, 2024.

[5] J. Zhang, C. Tan, Z. Cai, L. Zhu, Y. Feng, and S. Liang, "Cellular traffic forecasting based on inverted transformer for mobile perception dual-level base station sleep control," *Ad Hoc Networks*, vol. 161, Art. 103505, 2024.

[6] S. K. Mani, Y. Zhou, K. Hsieh, S. Segarra, T. Eberl, E. Azulai, I. Frizler, R. Chandra, and S. Kandula, "Enhancing network management using code generated by large language models," in *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*, 2023.

[7] Y. Zhou, K. Hsieh, S. K. Mani, S. Kandula, and Z. Liu, "MeshAgent: Enabling Reliable Network Management with Large Language Models," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 9, no. 3, Art. 52, Dec. 2025.

[8] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," arXiv:2210.03629, 2022.

[9] O. Khattab, M. Saad-Falcon, T. Li, E. Zhang, C. Wang, K. Mirchandani, and C. Potts, "DSPy: Compiling declarative language model calls into self-improving pipelines," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2024.

[10] A. F. Ansari, L. Stella, C. Turkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. Pineda Arango, S. Kapoor, J. Zschiegner, D. C. Maddix, M. W. Mahoney, K. Torkkola, A. G. Wilson, M. Bohlke-Schneider, and Y. Wang, "Chronos: Learning the language of time series," *Trans. Mach. Learn. Res.*, 2024.

[11] A. F. Ansari, O. Shchur, J. Küken, A. Auer, B. Han, P. Mercado, S. S. Rangapuram, H. Shen, L. Stella, X. Zhang, M. Goswami, S. Kapoor, D. C. Maddix, P. Guerron, T. Hu, J. Yin, N. Erickson, P. M. Desai, H. Wang, H. Rangwala, G. Karypis, Y. Wang, and M. Bohlke-Schneider, "Chronos-2: From univariate to universal forecasting," arXiv:2510.15821, 2025.

[12] G. Barlacchi,M. De Nadai,R. Larcher,A. Casella,C. Chitic,G. Medeossi, M. Noulas,S. P. Arango,A. Passarella,and B. Lepri, "A multi-source dataset of urban life in the city of Milan and the Province of Trentino," *Scientific Data*, vol. 2, p. 150055, 2015.