AegisLights: Self-Adaptive Signal Control Project Report

Group 8: Prabhjyot Singh, Yixing Zhang

I. Abstract

Urban cities still rely largely on fixed-time or locally actuated traffic signal plans that struggle under variable demand and unexpected disturbances (incidents, lane closures, events). Recent self-adaptive examples such as ATRP and TRAPP demonstrate adaptation through vehicle local decisions (individual or decentralized route choice), achieving lower travel time by steering drivers through the network. However, this project targets the infrastructure side of the problem. We treat traffic intersections themselves as the managed system and adapt traffic signals at runtime using a MAPE-K loop.

Our system adds a managing system to a simulated urban network. We model intersections as nodes in a graph and road segments as directed edges. The managing system monitors queue lengths, delays, and spillback to prevent intersection blocking, and adapts signal phases at cycle boundaries with rollback if performance degrades. This project extends beyond routine adaptation by being incident-aware: when a crash or blockage is detected, the affected edges are penalized in the cost model, causing the adaptation algorithm to favor signal phases that reduce flow through the incident area. Through simulation and data-driven adaptation using a contextual bandit algorithm, our goal is to demonstrate measurable reductions in average trip time. We hypothesize that infrastructure-level adaptation can reduce average trip time versus fixed-time baselines. We evaluate using a modified CityFlow micro-network across varying traffic scenarios, reporting average trip time over 120 adaptation cycles. By focusing on signals rather than routes, AegisLights complements approaches like ATRP/TRAPP and highlights infrastructure-based self-adaptation with runtime safety mechanisms, including performance-based rollback. Preliminary experiments show that the adaptive controller reduces average trip time by an average of 45% compared to fixed-time baselines under low, medium and high traffic conditions.

II. Introduction

Every city experiences highly variable demand on its roads during commute peaks, variable weather conditions and incidents such as collisions or stalled vehicles. Fixed time coordination only performs well near the load it was designed for, even reactive control is reactive and local to any given intersection. This design system often fails to prevent long queues at signals and queue spillback into intersections that further results in traffic problems. To solve this issue and achieve travel time reduction without costly infrastructure expansion, a self-adaptive approach may be the best option. It should sense emerging conditions, plan adjustments that follow safety/legal rules and continuously learn better responses to situations. The traffic rules for each intersection will vary according to local laws and regulations. These intersection-specific traffic rules, together with the directed graph representing the area map, will serve as inputs to our program.

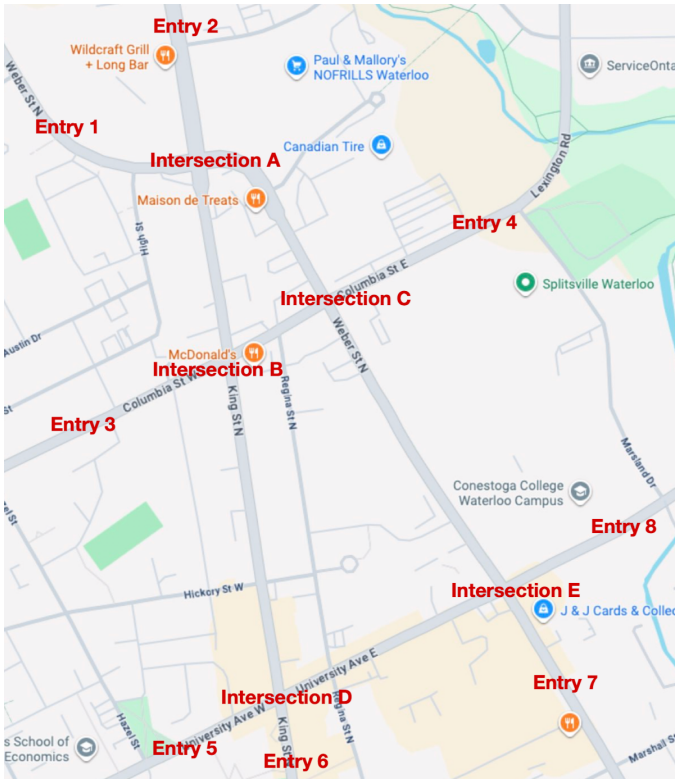

Fig. 1. Target intersections

A. Managed System and Adaptation Controls

The managed system for this project is an urban network of 5 signalized intersections connected by 28 directed road segments, implemented as a test network in the CityFlow simulator. Adaptation acts on signal phase selection at each intersection, determining which traffic movements receive green time at any given moment. The controller selects from a library of 4 predefined signal phases per intersection using a contextual bandit algorithm. This choice of control mechanism is deliberately different from vehicle-centric adaptation (e.g., ATRP/TRAPP) as we do not assign routes to cars and rather

aim to reshape priorities at junctions based on real-time traffic conditions.

B. Adaptation Goals

The primary adaptation goal is to minimize average trip time for vehicles under varying traffic demand scenarios. Secondary goals include minimizing the increase in travel time when incidents occur and reducing spillback events which pose safety concerns and further increase congestion.

C. Control & Safety Constraints

To avoid violating any laws and ensuring the safety of people when this system is active, some control and safety constraints must be put in place. Normal traffic laws should apply, and any given intersection should not have all green or red signals. Changes to the signal timings need to apply at cycle boundaries to avoid sudden changes. Lastly, there needs to be a sense to revert to a verified plan if performance indications show a degradation over a few cycles.

D. Research Questions

There are two main questions we want to answer through this project:

- **RQ1**: Can signal-level self-adaptation (MAPE-K) outperform fixed-time baselines on average trip time under varying demand?
- **RQ2**: How much does incident-aware planning reduce commute time versus incident-unaware adaptation?

E. Assumptions & Scope

For this project, some assumptions limit the scope. No direct vehicle control aside from selecting vehicles for simulated traffic accidents is applied, and as such benefits in travel time arise from signal phase prioritization only. Evaluation uses the CityFlow microsimulator with configurable traffic flow patterns (low, medium, high demand). The traffic represented in the simulator is limited to vehicles following predefined origin-destination routes.

III. Methodology

To develop a system targeting infrastructure self-adaptation rather than directing vehicle routing (as in ATRP and TRAPP, which optimize vehicle decisions), we adapt the signal network itself. We model a small urban grid of 5 signalized intersections and 28 road segments as a directed graph where nodes are intersections and edges are road approaches. The system streams relevant data from a simulator (queue lengths, delays, spillback indicators, incidents) and reweighs the graph each cycle to reflect congestion risk. A contextual bandit algorithm then selects signal phases that favor dynamically discovered bypass routes while penalizing congested edges. This shifts adaptation control from drivers to signals under legal/safety constraints. Our hypothesis is that infrastructure adaptation can reduce average trip time and improve incident response without controlling driver behavior. We use a Flask API written in Python to run the CityFlow engine and interface with the MAPE-K system, with lock-based concurrency to ensure synchronization.

A. CityFlow Simulator

The simulator was developed on top of the Flask framework and leverages CityFlow's API to expose a set of specialized endpoints tailored for integration with the MAPE-K feedback loop. Rather than providing only basic simulation control, the simulator API provides endpoints for receiving real-time traffic information, allowing the adaptive system to observe, analyze, and respond to evolving roadway conditions. These data streams include the number of vehicles waiting or moving along each road segment, aggregated mobility metrics such as average travel time, and explicit identification of vehicles currently designated as being in an accident state. To maintain consistency and correctness under concurrent access of the CityFlow engine, the simulator employs a lock-based synchronization mechanism that protects the traffic state during read or write operations. The lock is needed as there are multiple calls to the CityFlow engine in some API endpoints, so to avoid race conditions, the lock is held until all of an API endpoint's calls to the CityFlow engine are completed. This design ensures that each retrieved snapshot remains coherent and prevents race conditions, while also providing a degree of fault tolerance when the system experiences high request volume or delayed responses.

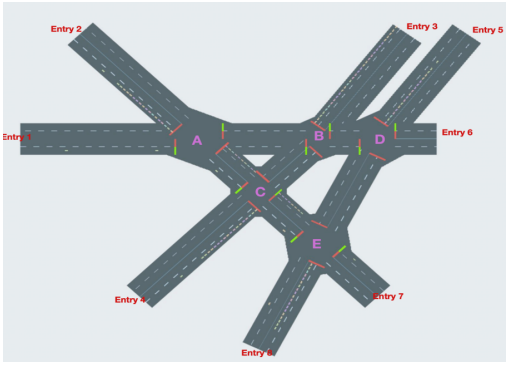

Fig. 2. Simulator Road Net

Internally, the simulator utilizes multithreading where the primary simulation loop operates concurrently with incoming API requests, all sharing a single CityFlow engine instance protected by a mutex lock. The simulation loop advances the simulation at configurable intervals (default 0.1 seconds real-time per simulation step, achieving 10x speedup) and introduces artificial accident events functionality not natively supported by CityFlow. To emulate incidents, the loop periodically selects a random vehicle (configurable interval via `accident_freq`, default every 900 simulation steps) and sets its speed to zero, simulating an immobilized vehicle. Each accident persists for 1800 simulation steps (aprox. 30 minutes of simulated time) before automatically clearing, and only one accident can be active at any time. Information about active accidents is stored in the simulator's internal state and exposed through API endpoints, enabling the MAPE-K system to detect and respond to these disturbances.

Because Flask creates a new thread to handle each incoming API request, the simulator may be accessed concurrently by multiple external components of the MAPE-K system. To ensure that interactions with the CityFlow engine remain safe and deterministic, all such interactions are guarded by a shared lock, whether they triggered by the main simulation loop or from API endpoints. Although this synchronization strategy does not guarantee that the response reflects the most immediate state of the simulation at the exact moment of the request, it ensures that the returned data is internally consistent and free from corruption. This design choice provides a balance between responsiveness and reliability, allowing the simulator to operate robustly under varying workloads while offering a stable interface for runtime adaptation and decision-making.

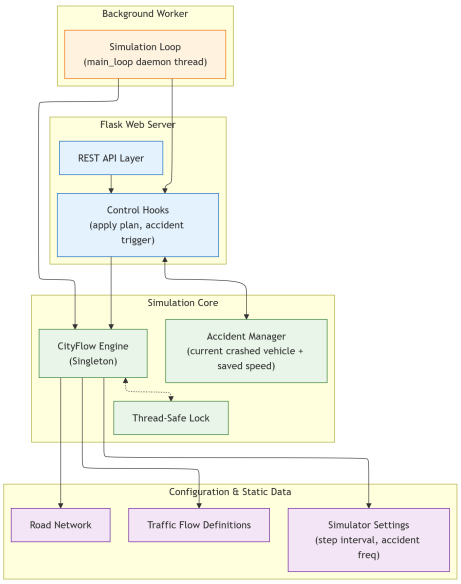

Fig. 3. Simulator Architecture

B. AegisLights Controller

The AegisLights controller implements a MAPE-K (Monitor-Analyze-Plan-Execute-Knowledge) feedback loop that continuously adapts traffic signal phases based on real-time conditions observed from the CityFlow simulator. The controller operates on a 3-second cycle, querying the simulator for traffic state, analyzing congestion patterns, selecting optimal signal phases using a contextual bandit algorithm, and executing changes via the simulator API. The system is designed around three core components: the MAPE-K adaptation loop, a SQLite database for persistent knowledge storage, and a web-based visualizer for real-time monitoring.

1) Graph Representation: The traffic network is modeled as a directed graph where nodes represent intersections and edges represent road segments. The network consists of 5 signalized intersections (A through E) and 8 virtual entry/exit nodes (1 through 8), connected by 28 directed edges. Each edge maintains runtime state including current queue length, delay, spillback status, incident flag, and a computed cost value. The graph model is updated each cycle with fresh data from the simulator and serves as the shared data structure across all MAPE-K components.

Edge costs are computed using a weighted formula that reflects congestion severity:

$$w_e(t) = \alpha \cdot \text{delay}_e + \beta \cdot \text{queue}_e + \gamma \cdot \text{spillback}_e + \delta \cdot \text{incident}_e \quad (1)$$

where the coefficients α, β, γ, and δ are configurable weights that prioritize safety-critical conditions. Spillback and incident penalties are set higher than delay and queue weights to ensure the system responds aggressively to blocking conditions that could cascade through the network.

2) MAPE-K Components: The adaptation loop executes four stages sequentially each cycle, with the Knowledge base providing persistent storage and retrieval across all stages.

a) Monitor: The Monitor stage collects the current traffic state from the CityFlow simulator via REST API. The simulator returns lane-level data including vehicle counts per lane, waiting vehicle counts (vehicles with speed below 0.1 m/s), and the current average travel time across all completed trips. This raw data is aggregated into edge-level metrics by summing across lanes belonging to each road segment. The monitor maintains a rolling window of the last 5 cycles to compute smoothed values, reducing noise from momentary fluctuations. Anomaly detection identifies two critical conditions: spillback (when queue length exceeds lane capacity, risking intersection blockage) and active incidents (vehicles marked as crashed by the simulator). The collected snapshot and detected anomalies are stored in the Knowledge base for use by subsequent stages. Figure 4 illustrates the workflow.

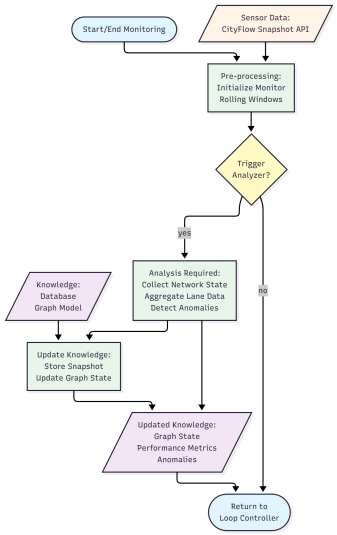

Fig. 4. Monitor Stage Workflow

b) Analyze: The Analyze stage processes monitored data to identify congestion patterns and determine which intersections require adaptation. Edge costs are recomputed using the weighted formula, and edges exceeding a configurable hotspot threshold (default 0.7) are flagged as congested. For

each hotspot, the analyzer uses Dijkstra's algorithm on the NetworkX graph representation to find k-shortest alternative paths (default $k = 3$) that bypass the congested edge. Exponential smoothing with parameter $\alpha = 0.3$ predicts near-term congestion trends, allowing the system to act proactively before conditions deteriorate. The analyzer also groups adjacent intersections for coordination, enabling synchronized phase changes along arterial corridors. The output includes prioritized adaptation targets, bypass routes, trend predictions, and any detected incidents with their affected edges. Figure 5 shows the analysis workflow.

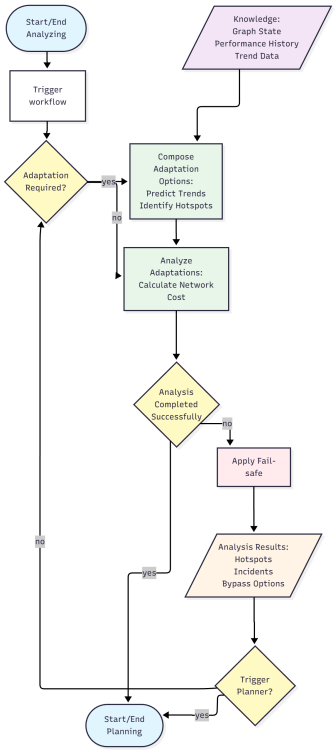

Fig. 5. Analyze Stage Workflow

c) Plan: The Plan stage selects signal phases for each intersection identified as needing adaptation. Each signalized intersection has 4 predefined phases (indexed 0–3) stored in a phase library, representing different configurations of which traffic movements receive green time. A contextual bandit algorithm balances exploitation of phases with known good performance against exploration of less-tested alternatives. By default, Upper Confidence Bound (UCB) selection is used, where for each candidate phase the planner computes:

$$\text{UCB}_a = \bar{r}_a + c \cdot \sqrt{\frac{\ln N}{n_a}} \tag{2}$$

where $\bar{r}_a$ is the average observed reward for phase a, N is total selections across all phases, n_a is selection count for phase a, and $c = 0.2$ is the exploration factor. The phase with highest UCB score is selected. Thompson Sampling is available as an alternative algorithm. In incident mode, when an active accident is detected, the planner applies additional cost penalties to phases that would increase flow toward affected edges, biasing selection toward phases that naturally divert traffic. Figure 6 details the planning process.

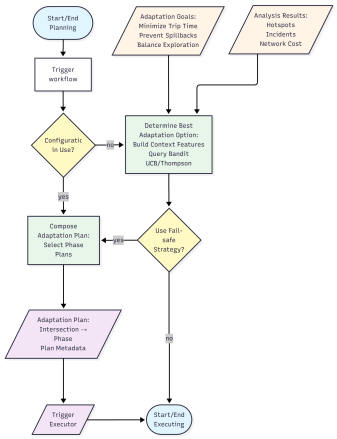

Fig. 6. Plan Stage Workflow

d) Execute: The Execute stage validates and applies the planned phase changes to the simulator. Safety validation ensures all phase IDs are within valid bounds (0–3), target only signalized intersections (not virtual entry/exit nodes), and correspond to entries in the phase library. Validated phases are applied via POST requests to the simulator's signal control endpoint, which calls CityFlow's `set_tl_phase()` function. After execution, performance metrics are calculated from the current graph state and compared against a rolling baseline of the previous 3 cycles. If the utility score (based on network cost) degrades by more than 10% compared to baseline, the executor triggers automatic rollback to the last known good configuration, providing runtime safety against poorly performing adaptations. Successfully applied configurations update the baseline. Finally, the bandit reward is calculated and stored:

$$r = -\text{avg\_trip\_time} - 50 \cdot \text{spillbacks} - 2 \cdot \text{avg\_queue} \tag{3}$$

This reward formulation directly optimizes for the primary goal (minimizing trip time) while heavily penalizing safety-critical spillback events and maintaining pressure to reduce queue buildup. Figure 7 shows the execution workflow.

3) Database: The SQLite database serves as the Knowledge component of the MAPE-K loop, providing persistent storage for both runtime state and historical analysis. The `simulation_snapshots` table records cycle-level traffic observations including queue lengths, delays, and incident flags for each edge. The `graph_state` table maintains the current state of each edge, serving as the runtime model that Monitor and Analyze stages query. Signal control actions are logged in `signal_configurations`, creating an execution trace linking decisions to outcomes. The `phase_libraries` table stores the 4 valid phases per intersection, functioning as the pre-validated action space for the planner. Evaluation data in `performance_metrics`

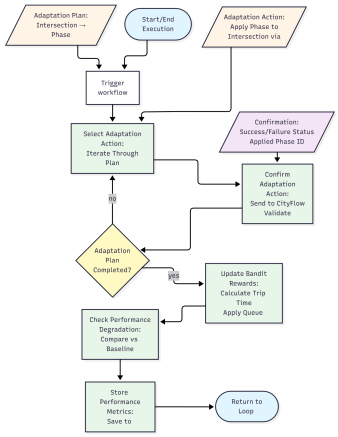

Fig. 7. Execute Stage Workflow

enables post-hoc analysis of controller effectiveness. Finally, `bandit_state` persists the contextual bandit's learning statistics, such as selection counts, cumulative rewards, and average rewards per phase, which enables the controller to resume learning across restarts.

4) Visualizer: The web visualizer provides a real-time dashboard for monitoring network state during experiments. Implemented as a standalone Flask server with a D3.js frontend, it reads directly from the SQLite database and renders an interactive force-directed graph. Edges are color-coded by severity (green for low cost, yellow for moderate, red for high cost/incidents), with line thickness indicating queue length. The dashboard displays current cycle number, active incident count, recent adaptation count, and average delay. Trend charts show the evolution of key metrics over the last 50 cycles. The visualization auto-refreshes every 5 seconds, synchronized with the controller's adaptation cycle. This tool supports debugging, demonstration, and validation of adaptation behavior without interfering with the autonomous MAPE-K loop. Figure 8 shows the interface.

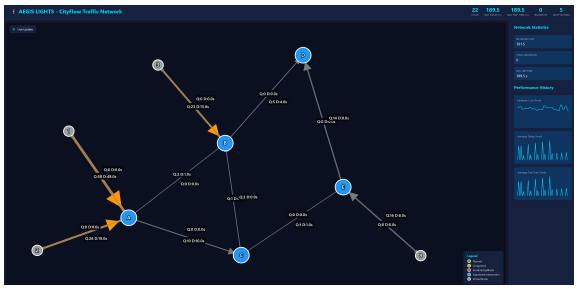

Fig. 8. AegisLights Web Visualizer

IV. Obtained Results

To evaluate the effectiveness of the MAPE-K adaptation framework, we conducted controlled experiments using the CityFlow traffic simulator across varying traffic conditions. Each experiment compared two configurations: a **control baseline** using fixed-time signal phases with no adaptation, and an **experimental condition** with the full MAPE-K pipeline enabled. Both configurations ran under identical starting conditions (same random seed, network topology, and traffic flows) for 120 adaptation cycles (360 seconds), allowing direct comparison of average travel time as the primary performance metric.

A. Experiment 1: Varying Traffic Loads

The first experiment evaluated system performance across three traffic demand levels by varying the vehicle spawn interval in the flow configuration. The low traffic scenario used a 30-second spawn interval per route, producing approximately 24 vehicles per minute across all entry points. The medium traffic scenario reduced this to a 10-second interval, yielding roughly 72 vehicles per minute. The high traffic scenario used a 4-second interval, generating approximately 180 vehicles per minute, a load designed to stress the network beyond comfortable capacity.

Table I summarizes the results across all three conditions.

TABLE I
Performance Comparison Across Traffic Load Levels

Metric	Low	Medium	High
Control Mean (s)	78.56	409.47	679.24
Experimental Mean (s)	40.05	219.09	356.63
Improvement	49.02%	46.49%	47.50%
Control Std Dev	12.25	213.80	384.57
Experimental Std Dev	40.19	266.45	450.29

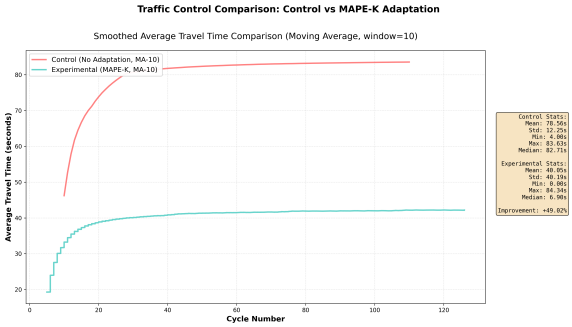

Fig. 9. Low Traffic Scenario: Control vs MAPE-K Adaptation

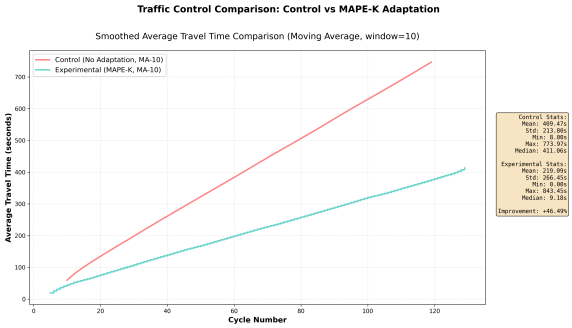

Fig. 10. Medium Traffic Scenario: Control vs MAPE-K Adaptation

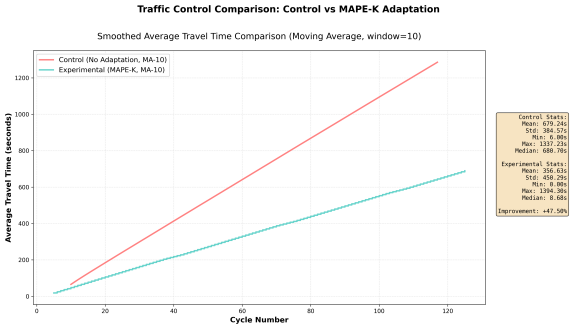

Fig. 11. High Traffic Scenario: Control vs MAPE-K Adaptation

a) Analysis: The MAPE-K controller achieved consistent improvements of approximately 46–49% reduction in average travel time across all traffic load levels. The stability of this percentage improvement regardless of absolute traffic volume suggests that the adaptive mechanism scales effectively with demand. This indicates the contextual bandit successfully learns phase selections that outperform fixed-time defaults across varying conditions, rather than being tuned for a specific load level.

In the low-traffic scenario illustrated in Figure 9, both curves plateau after approximately 40 cycles, indicating the network reached steady-state conditions where vehicle arrivals and departures balanced. The control baseline stabilized at approximately 83 seconds average travel time while the adaptive system maintained roughly 42 seconds. The relatively flat curves demonstrate that under light load, the network is not capacity-constrained; however, the adaptive controller still identifies more efficient phase configurations that reduce unnecessary waiting at signals.

The medium and high traffic scenarios shown in Figures 10 and 11 exhibit fundamentally different behavior. Travel times grow continuously throughout the experiment as vehicles accumulate in the network faster than they can exit. Critically, the rate of this growth differs substantially between the two systems. Under high traffic conditions, the control baseline reaches 1337 seconds by cycle 120, while the adaptive system reaches only 700 seconds. This divergence demonstrates that adaptation prevents the exponential congestion buildup characteristic of saturated fixed-time systems, where small delays compound as queues propagate upstream through the network.

The experimental condition exhibits higher standard deviation than the control in some scenarios, which reflects the bandit's exploration behavior. The algorithm occasionally selects suboptimal phases to gather reward information about less-tested alternatives. However, the substantially lower mean travel time indicates this exploration cost is outweighed by the exploitation of discovered better configurations over the course of the experiment.

B. Experiment 2: Biased Traffic with Incident Response

The second experiment tested system resilience under stress conditions using a biased traffic pattern that concentrated demand at a single entry point, with and without simulated accidents. This scenario was designed to evaluate both the system's ability to handle asymmetric demand and its incident response capabilities.

The biased flow pattern spawned vehicles at 2-second intervals from entry node 1 (via road 1A) across five different routes, while other entry points used 8-second intervals. This configuration directed approximately 67% of total traffic through intersection A, simulating a rush-hour corridor or event-driven demand surge where one arterial experiences disproportionate load.

When enabled, accidents were injected every 900 simulation steps, corresponding to approximately 15 minutes of simulated time. Each accident immobilized a randomly selected vehicle for 1800 steps (approximately 30 minutes of simulated time), creating a moving bottleneck that propagated delays through the network as following vehicles queued behind the obstruction.

Table II compares the biased traffic scenarios with and without incident injection.

TABLE II
PERFORMANCE UNDER BIASED TRAFFIC LOAD

Metric	No Accident	With Accident
Control Mean (s)	714.65	712.38
Experimental Mean (s)	394.90	382.92
Improvement	44.74%	46.25%
Control Max (s)	1408.42	1407.22
Experimental Max (s)	1595.03	1496.77

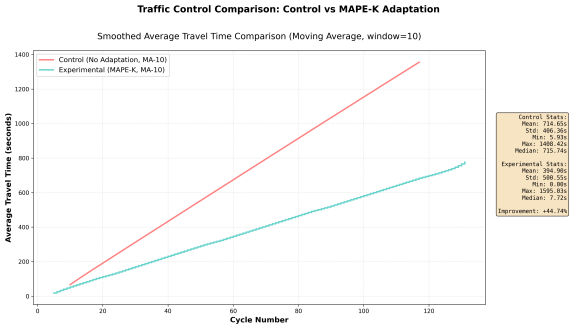

Fig. 12. Biased Traffic Without Accidents: Control vs MAPE-K

a) Analysis: Under the biased traffic configuration, the adaptive controller achieved 44–46% improvement despite the severe demand imbalance. The contextual bandit learned to favor phases that prioritized the overloaded corridor through intersection A while maintaining acceptable service levels on lower-demand approaches. Fixed-time control, by contrast, allocated equal green time to all approaches regardless of actual demand, leading to unnecessary delays on the high-volume routes while underutilizing capacity on quieter approaches.

Comparing the accident-enabled scenario to the no-accident baseline reveals a counterintuitive result: the MAPE-K system achieved slightly better relative performance (46.25% improvement) when incidents occurred than without incidents (44.74% improvement). This suggests the incident-aware

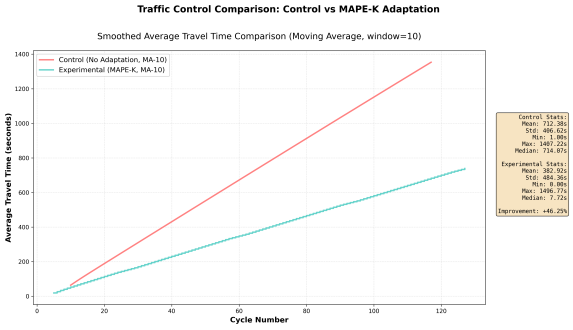

Fig. 13. Biased Traffic With Accidents: Control vs MAPE-K

planning mechanism successfully detected blocked edges and biased phase selection away from affected routes. The edge cost formula's incident penalty term increases costs on affected edges, causing the analyzer to identify bypass routes and the planner to select phases that reduce flow toward the blockage rather than continuing to feed traffic into a bottleneck.

The control baseline showed nearly identical performance with and without accidents, achieving mean travel times of 714.65 seconds and 712.38 seconds respectively. This similarity indicates that under already-saturated conditions, random accidents have minimal additional impact on a system that is already failing to clear traffic efficiently. The fixed-time controller has no mechanism to respond to incidents, but the network was already congested enough that the marginal effect of an additional blockage was absorbed into the existing delays. The adaptive system, by maintaining lower baseline congestion throughout the experiment, retained more capacity headroom to absorb incident-induced delays without catastrophic performance degradation.

The experimental condition showed higher maximum travel times than control in some runs, such as 1595 seconds versus 1408 seconds in the no-accident biased scenario. These outliers likely correspond to vehicles that entered the network during bandit exploration phases when suboptimal signal configurations were being tested, or vehicles that were caught in transient congestion before the system had time to adapt. The significantly lower mean and median values indicate these are rare events affecting a small number of trips rather than systematic failures of the adaptive mechanism.

C. Summary of Findings

Across all five experimental conditions, the MAPE-K adaptive controller demonstrated consistent and substantial performance improvements over the fixed-time baseline. The system achieved 44–49% reduction in average travel time across scenarios ranging from light traffic to heavily biased demand with active incidents.

The performance gains remained stable from low to high traffic loads, indicating that the bandit-based planning mechanism adapts effectively to varying demand levels rather than being optimized for a narrow operating range. Under high-load conditions where fixed-time control leads to runaway queue growth, the adaptive system maintained a substantially lower rate of travel time increase by continuously adjusting phase selections to match evolving congestion patterns.

The incident-aware planning mechanism demonstrated its value by maintaining or improving relative performance when accidents were introduced. While the absolute travel times increased for both systems when incidents occurred, the adaptive controller's ability to detect affected edges and bias phase selection toward bypass routes prevented the cascading delays that would otherwise propagate through the network.

These results provide an affirmative answer to **RQ1**: signal-level MAPE-K adaptation significantly outperforms fixed-time baselines on average trip time under varying demand conditions. For **RQ2**, the comparable or improved performance under incident conditions suggests that incident-aware planning provides measurable benefit. The effect appears less dramatic than the overall adaptation benefit, likely because the biased traffic scenario was already near saturation, limiting the headroom available for incident-specific optimization. Further experiments under moderate-load conditions with incidents would better isolate the contribution of incident awareness.

D. Limitations

Several limitations constrain the generalizability of these results. All experiments used the same 5-intersection synthetic network, leaving performance on larger or differently-structured networks untested. Real urban networks exhibit more complex topology, including varying block lengths, turn prohibitions, and coordination requirements across dozens or hundreds of intersections.

The experiments used a fixed random seed for reproducibility, enabling direct comparison between control and experimental runs. However, establishing statistical significance would require multiple runs with varying seeds to account for stochastic variation in vehicle generation and routing. The reported improvements should therefore be interpreted as point estimates rather than statistically validated effects.

CityFlow's average travel time metric returns a cumulative average across all completed trips since simulation start, which may mask transient performance variations. A vehicle experiencing severe delay early in the simulation contributes to the average throughout, potentially obscuring improvements made in later cycles. Per-cycle or per-vehicle metrics would provide finer-grained insight into adaptation dynamics.

Finally, all results are based entirely on simulation. Real-world traffic exhibits additional complexities including driver behavior variation, pedestrian interactions, weather effects, and sensor noise that are not captured in this controlled environment. Validation against real-world deployments or higher-fidelity simulators would strengthen confidence in the approach's practical applicability.

V. Conclusion

AegisLights demonstrates that a signal-centric, MAPE-K–driven approach to urban traffic management can meaningfully improve network performance without exerting direct

control over driver routing. By modeling the traffic network as a directed graph with intersections as nodes and road segments as edges, and implementing a feedback loop that continuously monitors queue lengths and delays, analyzes congestion patterns, selects signal phases using a contextual bandit algorithm, and executes changes with rollback protection, the system achieves substantial reductions in average travel time using only infrastructure-level adaptations. Our experiments across five scenarios: spanning low, medium, and high traffic loads as well as biased demand patterns with and without simulated accidents, showed consistent improvements of 44–49% in average travel time compared to fixed-time signal control.

The experimental results provide affirmative answers to both research questions posed at the outset. For RQ1, the MAPE-K controller significantly outperformed fixed-time baselines across all tested demand levels, with the performance gap widening under high-load conditions where fixed-time control leads to runaway congestion while the adaptive system maintains controlled queue growth. For RQ2, the incident-aware planning mechanism demonstrated measurable benefit: when accidents were introduced in the biased traffic scenario, the adaptive system maintained or slightly improved its relative performance advantage by detecting affected edges and biasing phase selection toward bypass routes.

Beyond raw performance metrics, the project highlights the importance of a robust architectural foundation for self-adaptive systems. The CityFlow-backed simulator with its Flask API and lock-based synchronization provided a stable interface for real-time observation and control. The SQLite database schema enabled persistent storage of traffic snapshots, adaptation decisions, and bandit learning statistics, supporting both runtime operation and post-hoc analysis. The web-based visualizer offered real-time insight into network state and adaptation behavior. Together, these components enabled repeatable experimentation, detailed introspection, and end-to-end traceability of why specific phase selections were made.

The system's safety mechanisms proved essential for reliable operation. The 10% performance degradation threshold with automatic rollback to last-known-good configurations prevented poorly-performing adaptations from persisting. The phase library constraint ensured only pre-validated signal configurations could be applied, and the validation layer rejected any adaptations targeting virtual nodes or using invalid phase indices. These guardrails illustrate how safety and transparency can be embedded into adaptive traffic control without sacrificing responsiveness.

The primary learning outcome from this project is that infrastructure-level adaptation, supported by lightweight graph-based analytics, online learning through contextual bandits, and real-time sensing, can complement driver-level routing systems such as ATRP and TRAPP. Simultaneously, adaptation can avoid reliance on vehicle cooperation or navigation system adoption. The MAPE-K pattern provided a clear separation of concerns that facilitated independent development and testing of each stage, while the shared knowledge base enabled coherent decision-making across the loop.

Several limitations remain for future work. The current implementation operates on a small 5-intersection synthetic network; scaling to larger grids would require more sophisticated coordination mechanisms and potentially hierarchical control structures. The experiments used a single random seed, so statistical validation across multiple runs would strengthen confidence in the reported improvements. Additionally, the contextual bandit currently uses relatively simple context features; richer representations incorporating predicted demand, time-of-day patterns, or learned congestion dynamics could further improve adaptation quality.

In summary, AegisLights provides a proof-of-concept that infrastructure-level self-adaptation can achieve meaningful traffic performance improvements while maintaining safety constraints and decision transparency. The consistent 46–49% reduction in average travel time across diverse scenarios establishes a practical foundation for future research into scalable, interpretable, and responsive traffic control systems.

REFERENCES

[1] I. Gerostathopoulos and E. Pournaras, "TRAPPed in Traffic? A Self-Adaptive Framework for Decentralized Traffic Optimization," in *Proceedings of the 2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2019, pp. 32–38. [Online]. Available: https://ieeexplore.ieee.org/document/8787057

[2] J. Wuttke, Y. Brun, A. Gorla, and J. Ramaswamy, "Traffic routing for evaluating self-adaptation," in *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '12)*, Zurich, Switzerland, 2012, pp. 27–32. [Online]. Available: https://www.hpi.uni-potsdam.de/giese/public/selfadapt/exemplars/model-problem-atrp/

[3] H. Zhang, S. Feng, C. Liu, Y. Ding, Y. Zhu, Z. Zhou, W. Zhang, Y. Yu, H. Jin, and Z. Li, "CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario," in *Proceedings of the 2019 World Wide Web Conference (WWW '19)*, San Francisco, CA, USA, 2019, pp. 3620–3624. [Online]. Available: https://github.com/cityflow-project/CityFlow/

[4] F. Montazeri and N. Frejinger, "Adaptive Traffic Signal Control: A Review of Methods and Applications," M.S. thesis, École de technologie supérieure, Montreal, QC, Canada, 2023. [Online]. Available: https://espace.etsmtl.ca/id/eprint/3220/1/MONTAZERI_Farzaneh.pdf

[5] P. Singh and Y. Zhang, "AegisLights: Self-Adaptive Traffic Signal Control Using MAPE-K," Project Proposal, ECE 750: Self-Adaptive Systems, University of Waterloo, Waterloo, ON, Canada, 2025.

[6] P. Singh and Y. Zhang, "AegisLights: MAPE-K Traffic Signal Controller with CityFlow Simulator," GitHub Repository, 2025. [Online]. Available: https://github.com/Prabhjyot045/aegis_lights