Agentic Compute–Energy Co-Optimization System Using Beckn Protocol

MVP Prototype and Extensible Design

1 Background and Motivation

AI workloads such as training, inference, and batch processing are rapidly growing into major sources of electricity demand. At the same time, electricity systems are becoming more volatile and complex because carbon intensity varies hour by hour, prices respond dynamically to renewable output, and networks face local congestion. This combination creates a new coordination challenge: compute is rising as a large, flexible load, while grids are increasingly shaped by distributed renewables and real-time constraints.

In this setting, compute jobs can be treated as flexible and shiftable energy loads. Many AI tasks do not need to run immediately in a fixed location. Instead, they can be deferred within bounded windows, shifted across regions, paused and resumed, or scheduled to coincide with low-carbon and low-price electricity periods. If these flexibilities are used systematically, it becomes possible to co-optimise the direct monetary cost of compute (for instance, £ per inference), the carbon footprint of compute, and the operational stress placed on local grids. This framing aligns with the Digital Energy Grid vision, which argues that future energy systems require a unified digital layer to coordinate information, energy, and transactions across decentralized actors.

However, today compute scheduling is mostly grid-agnostic. Cluster schedulers and cloud orchestration tools rarely incorporate real-time carbon signals, congestion warnings, or flexibility incentives. Conversely, grid-side flexibility platforms are typically unaware of compute workload structures such as duration, preemption risk, deferrability, and geographic portability. The result is that compute flexibility is left unused, while grids are forced to balance supply and demand with limited visibility into one of the fastest-growing loads.

Our project proposes a minimal but functional agentic orchestration system that bridges these worlds. We use the Beckn Protocol as the interoperability layer to enable a standard transaction workflow between compute agents and grid agents. A compute agent publishes flexible job slots as discoverable catalog items, a grid agent evaluates these slots against evolving grid conditions, and an orchestration engine selects the best feasible time and place to execute each task. This "search → select → confirm" lifecycle provides a practical foundation for compute–energy convergence in a DEG-aligned ecosystem.

2 Problem Statement

The core problem is to determine when and where to run compute tasks so that the overall cost of compute is minimized, a carbon intensity cap is respected, workload constraints are met, and execution aligns with grid conditions. These grid conditions include not only carbon intensity and price but also local congestion, renewable availability, and explicitly signalled flexibility incentives. In practice, this requires balancing multiple objectives and constraints that are currently represented in different and non-interoperable systems.

There is no unified mechanism today through which compute agents can publish flexible job slots with standardized semantics, and grid agents can evaluate and confirm those slots using shared digital rules. The Digital Energy Grid paper identifies this kind of fragmentation as a structural barrier to resilient, scalable coordination, and emphasizes three universal digital building blocks needed for broad interoperability: universal identity for assets and actors, machine-readable data formats for all relevant descriptors, and verifiable digital records for trusted transactions. Our proposal therefore focuses on designing and implementing an agentic compute–energy orchestration system that embodies these three principles in an MVP form: compute agents publish job slots as Beckn catalog items with unique asset identities and machine-readable descriptors, grid agents confirm or reject slots through a Beckn-style order lifecycle using a structured information plane, and a lightweight orchestration engine performs co-optimisation under simplified but realistic constraints.

3 Objectives (MVP Scope)

Given the tight time budget of one day for proposal writing and one day for implementation, we define an MVP that demonstrates the essential DEG-aligned coordination loop while

keeping the design extensible.

The compute agent models a small catalogue of representative compute tasks, such as inference batches and training runs. Each task is represented as a flexible digital asset rather than a simple job request. Alongside duration, deferral windows, region options, and priority, tasks include a globally unique `compute_asset_id`, a `location_id` for each permissible region, an energy profile with estimated `power_kw` and ramp characteristics, and a flexibility class that indicates whether the task is merely deferrable or also interruptible or migratable. The compute agent expands tasks into candidate job slots and publishes these slots through a Beckn-style `/search` endpoint as catalog items enriched with those semantics.

The grid agent simulates time-varying grid signals. In addition to carbon intensity, price, and congestion flags, it includes renewable share and a simple flexibility incentive signal representing the grid value of load shifting at each hour. These signals are grouped into an "information plane" representation so that orchestration decisions can draw from a coherent digital view of the grid rather than isolated metrics. During confirmation, the grid agent evaluates selected slots against the carbon cap, any price or congestion thresholds, and the presence of flexibility incentives, then returns a confirmation or rejection via a Beckn-style `/confirm` endpoint.

The orchestration engine performs co-optimisation. In the MVP it uses an interpretable rule-based scoring method to rank candidate slots by a weighted combination of carbon intensity and energy price, while also accounting for flexibility incentives where available. An optional LLM component acts as an advisory meta-agent that can explain decisions or suggest trade-offs, but final decisions remain deterministic and auditable.

The Beckn workflow is implemented through `/search`, `/select`, and `/confirm`. Although simplified, the lifecycle mirrors Beckn's transaction model and is explicitly designed to scale toward a fuller order flow with asynchronous callbacks and status endpoints.

Finally, the MVP maintains auditability as a first-class objective. For every decision, the system stores a structured audit record containing task identity, slot choice, grid state, objective values, and reasons. To align with the DEG verifiability pillar, audit entries are chained by hashes so that the log becomes tamper-evident, and the structure is compatible with later addition of cryptographic signatures.

3.1 Explicit Non-Goals for MVP

To keep the MVP achievable in one implementation day, we will not integrate with real GPU clusters, live grid APIs, advanced optimisation algorithms such as MILP or reinforcement learning, or a production-grade UI. Instead, we rely on realistic simulations and focus on the orchestration logic, standardised protocol interactions, and DEG-aligned digital representation of assets and transactions.

4 System Architecture (MVP)

4.1 High-Level Components

The MVP is implemented as three logical components orchestrated by a single FastAPI server. The compute agent stores and exposes the compute task catalogue, generates job slots, and publishes them through `/search` while accepting selections through `/select`. The grid agent stores the simulated information plane over time and enforces confirmation logic through `/confirm`. The orchestration engine reads task descriptors and grid signals, expands tasks into feasible slots, ranks them using the co-optimisation rule, and passes the selected slot forward for confirmation. Although the MVP uses one compute agent and one grid agent, the architecture is deliberately written as a "network of networks" template, so that multiple regional compute and grid agents can be added later without redesign.

4.2 Proposed Project Structure (Python)

```
project/
  app.py                    # FastAPI entry point (server and routing)
  compute_agent/
    tasks.json              # Compute task catalog with asset IDs and energy profiles
    compute_service.py      # /search and /select logic
  grid_agent/
    info_plane.json         # Simulated grid information plane (carbon, price, renewable
    grid_service.py         # /confirm logic
  orchestrator/
    scheduler.py            # Co-optimisation engine + optional LLM advisory
  logs/
    audit.json              # Hash-chained audit log of decisions
```

5 Methodology

5.1 Compute Task Modelling

We define example compute workloads in `tasks.json`. Each task includes a unique `id` for internal reference and a globally unique `compute_asset_id` to align with universal identity. It also specifies `duration_hours`, `deferrable_hours`, `region_options` tied to explicit `location_id`s, and `priority`. To make tasks machine-readable digital assets in the DEG sense, each entry includes an energy profile describing estimated `power_kw`, whether ramp-up is instantaneous or gradual, and a flexibility class that indicates if the task is strictly deferrable, interruptible, or migratable across regions. The compute agent loads this catalogue, expands each task into candidate job slots within its allowable deferral window and region set, and exposes those slots as a Beckn catalog.

5.2 Grid Information Plane Simulation

We simulate grid conditions in `info_plane.json` as a time series indexed by hour. For each hour and region, the information plane provides `carbon_intensity` in gCO_2/kWh, `price` in £/kWh, a `renewables_share` signal, an optional `congested` flag indicating network stress, and a simple `flex_reward` representing grid-provided value for load shifting into that hour. This grouping reflects the DEG emphasis on synchronising information flows with physical energy flows through coherent digital structures. During confirmation, the grid agent queries the relevant hour and location, applies the carbon cap and operational thresholds, and factors in renewable share and flexibility rewards as part of its decision and explanation.

5.3 Orchestration Engine and Optimisation Logic

The orchestration engine in `scheduler.py` encapsulates the co-optimisation logic. For each task, it enumerates candidate slots implied by the task's deferral window and region options, looks up grid conditions from the information plane for each slot, and computes a score of the form

$$\text{score} = w_1 \cdot \text{carbon\_intensity} + w_2 \cdot \text{price} - w_3 \cdot \text{flex\_reward},$$

where w_1, w_2, and w_3 are configurable weights that let us explore different trade-offs between carbon, cost, and grid value. The engine selects the candidate with minimum score that

respects the global carbon cap and avoids hard-congested intervals when possible. If no slot satisfies the cap, it selects the lowest-carbon feasible slot and records the constraint violation explicitly in the audit log. This provides a lightweight approximation of multi-objective optimisation suitable for an MVP while remaining transparent and extensible.

As an optional enhancement, we integrate an LLM as an advisory component. The model receives task attributes, candidate slots with their grid states, and the declared optimisation goals, then returns a suggested hour and region together with a short natural-language rationale. The system uses this rationale to enrich audit logs, but the deterministic rule remains the final authority.

5.4 Beckn-Style Workflow Implementation

We follow a simplified Beckn-inspired lifecycle. The compute agent exposes `/search`, which returns a Beckn-style catalog of job slots, each slot carrying identity, location, energy profile, flexibility class, and timing attributes in a machine-readable format. The compute agent also exposes `/select`, which accepts a task identifier and either records a client-chosen slot or automatically selects the best slot using the orchestration engine. The grid agent exposes `/confirm`, which receives the selected slot, validates it against the information plane and system constraints, and returns a final confirmation or rejection along with a reason grounded in carbon, price, congestion, and incentive signals. The workflow creates an explicit transaction boundary: decisions are not merely internal scheduling choices but verifiable digital exchanges between two agents.

5.5 Audit Logging and Verifiability

For every scheduling attempt, the system writes a structured JSON record to `logs/audit.json`. Each record includes the task's internal `id` and global `compute_asset_id`, the chosen hour and location, the full information-plane state at that slot, the computed score components, the confirmation outcome, and a human-readable justification. To support verifiability, each audit entry contains a cryptographic hash of its contents and the hash of the previous entry, forming a simple hash chain that makes tampering evident. This structure is compatible with later addition of digital signatures or external attestation, reflecting the DEG requirement for trusted, portable records.

6 Implementation Plan (2-Day Schedule)

On Day 1, we clarify scope, assumptions, and evaluation metrics, then design task and information-plane schemas that include asset identities, energy profiles, and flexibility semantics. We define Beckn-style request and response models for `/search`, `/select`, and `/confirm`, specify orchestration rules, and refine this proposal.

On Day 2, we implement the FastAPI server and the three components. We populate `tasks.json` with a small but realistic workload set and `info_plane.json` with simulated regional carbon, price, renewable share, congestion, and flexibility reward signals. We implement the rule-based scheduler and optional LLM hooks, then connect the full Beckn lifecycle end to end. Finally, we validate behavior through FastAPI's `/docs` interface and small scripted calls, and package the MVP with a short README for reproducible execution.

7 Future Extensions

Beyond the MVP, the system is intentionally designed to scale along multiple dimensions that align with both Beckn and DEG.

First, the Beckn lifecycle can be expanded to include asynchronous callbacks such as `/on_search` and `/on_select`, initiation and completion phases such as `/init` and `/on_confirm`, and operational endpoints such as `/status` and `/cancel`. This would allow long-running jobs, pauses, and cancellations to be represented as first-class protocol states.

Second, we plan an interactive dashboard that visualises the evolving information plane, the queue of tasks, the selected and confirmed slots, and aggregate cost and carbon savings. A lightweight web UI or Streamlit interface is sufficient for this stage and would make system behavior more interpretable to grid and compute stakeholders.

Third, the architecture supports multi-agent coordination. We can introduce multiple regional compute agents publishing heterogeneous catalog items and multiple grid agents representing different system operators. Slot selection could then follow negotiation, auction, or cooperative planning, showing clear alignment with the "network of networks" concept in the DEG paper.

Fourth, the rule-based score can be replaced or augmented with advanced optimisation approaches such as MILP, constraint programming, reinforcement learning, or Pareto-based multi-objective solvers. The orchestration engine is isolated precisely so that these upgrades do not require changes to protocol or agent logic.

Finally, we can integrate real data and infrastructure. This includes live carbon-intensity and electricity-price feeds, real congestion indicators, and direct integration with cluster schedulers or cloud APIs. These steps would convert the MVP into a realistic experimental platform for compute–energy co-optimisation in operational settings.

8 Expected Impact

Even in MVP form, the system demonstrates that compute workloads can be represented as flexible digital energy assets, that grid-aware scheduling can reduce both cost and carbon exposure, and that Beckn Protocol provides a credible interoperability rail for agent-to-agent coordination. By adding universal identities, machine-readable energy and flexibility descriptors, and verifiable audit records, the design also serves as a concrete example of how DEG principles can be instantiated in a compute–energy convergence use case. Over time, these capabilities can enable data centers and AI workloads to participate in flexibility markets, respond autonomously to renewable availability, and act as cooperative partners in maintaining grid stability, rather than as unmanaged sources of rising demand.

9 Conclusion

This proposal presents an MVP design for an agentic compute–energy co-optimisation system built on top of Beckn-style workflows and explicitly aligned with the Digital Energy Grid vision. The MVP is scoped for rapid implementation while embedding the DEG pillars of universal identity, machine-readable descriptors, and verifiable records. By connecting compute agents and grid agents through a shared protocol and a lightweight orchestration layer, the system illustrates a practical path toward grid-synchronised compute infrastructure, where AI workloads run not only when requested, but when and where the energy system can support them most sustainably and efficiently.