**راستی رفیق یک بار دیگه هر آنچه از چارچوب پیشنهادی SkyMind مخفف عبارت**

**System for Knowledge‑aware Mobility and Intelligent Network Decision‑making می دونی بگو و یک نقشه راه جامع برای پیاده سازی و شبیه سازی (با رویکرد مدل سازی انتزاعی در پایتون) و تهیه یک داشبورد تعاملی بیار.**

عالی! با کمال میل. این یک درخواست فوق‌العاده جذاب و استراتژیک است. بیایید با هم به اعماق چارچوب **SkyMind (System for Knowledge‑aware Mobility and Intelligent Network Decision‑making)** شیرجه بزنیم، آن را کالبدشکافی کنیم و سپس یک نقشه راه جامع برای پیاده‌سازی، شبیه‌سازی و ساخت داشبورد برای آن طراحی کنیم.

**بخش اول: بازخوانی و تحلیل عمیق چارچوب SkyMind**

همانطور که از نام کامل آن پیداست، SkyMind یک چارچوب مفهومی برای ساخت سیستم‌های هوشمندی است که در تقاطع سه حوزه کلیدی عمل می‌کنند:

1. **Mobility (پویایی/تحرک):** این بخش به هر نوع موجودیت متحرک اشاره دارد. این موجودیت‌ها می‌توانند فیزیکی (مانند خودروهای خودران، پهپادها، ربات‌های لجستیک، انسان‌ها با گوشی‌های هوشمند) یا مجازی (مانند بسته‌های داده در یک شبکه، وظایف پردازشی در یک کلاستر) باشند. هدف، درک، پیش‌بینی و بهینه‌سازی حرکت این موجودیت‌هاست.
2. **Intelligent Network Decision-making (تصمیم‌گیری هوشمند در شبکه):** این بخش بر روی “شبکه” به عنوان بستر ارتباطی و تعاملی این موجودیت‌های متحرک تمرکز دارد. تصمیم‌گیری‌ها می‌تواند شامل موارد زیر باشد:

* **مسیریابی بهینه:** پیدا کردن بهترین مسیر برای وسایل نقلیه یا بسته‌های داده.
* **تخصیص منابع:** اختصاص پهنای باند، توان محاسباتی (Edge Computing) یا ایستگاه‌های شارژ به موجودیت‌ها.
* **مدیریت ترافیک:** جلوگیری از ازدحام در شبکه‌های حمل‌ونقل یا شبکه‌های کامپیوتری.
* **تحویل محتوا:** تصمیم‌گیری برای اینکه کدام داده در کدام گره لبه (Edge Node) ذخیره (Cache) شود تا به سرعت به کاربران متحرک تحویل داده شود.

1. **Knowledge-aware (آگاه از دانش):** این "چسب"ی است که دو بخش دیگر را به هم متصل کرده و آن‌ها را هوشمند می‌سازد. سیستم صرفاً بر اساس داده‌های آنی و لحظه‌ای تصمیم نمی‌گیرد، بلکه از یک “پایگاه دانش” (Knowledge Base) استفاده می‌کند. این دانش می‌تواند شامل موارد زیر باشد:

* **دانش ایستا (Static):** نقشه جاده‌ها، توپولوژی شبکه، مشخصات فنی وسایل نقلیه، قوانین راهنمایی و رانندگی.
* **دانش پویا (Dynamic):** الگوهای ترافیکی تاریخی (مثلاً ترافیک در ساعات اوج)، شرایط آب و هوایی، رویدادهای در حال وقوع (کنسرت، تصادف).
* **دانش استنتاجی (Inferred):** دانشی که توسط خود سیستم و با استفاده از مدل‌های یادگیری ماشین تولید می‌شود. برای مثال، پیش‌بینی مناطق پرخطر برای تصادف یا پیش‌بینی تقاضا برای سرویس تاکسی در یک منطقه خاص.

**هدف نهایی SkyMind:** ایجاد یک “مغز متفکر دیجیتال” که با درک عمیق از محیط، تحرک موجودیت‌ها و دانش زمینه‌ای، بتواند تصمیمات بهینه، پیش‌بینانه و آنی را برای کل اکوسیستم شبکه و تحرک اتخاذ کند.

**بخش دوم: نقشه راه جامع برای پیاده‌سازی و شبیه‌سازی**

این نقشه راه، رویکرد **مدل‌سازی انتزاعی در پایتون** را دنبال می‌کند. یعنی به جای درگیر شدن با جزئیات فیزیکی بسیار پیچیده، ما اجزای سیستم را به صورت کلاس‌ها و اشیاء پایتونی مدل می‌کنیم تا بتوانیم منطق و تعاملات آن‌ها را شبیه‌سازی کنیم.

**فاز ۱: طراحی و پیاده‌سازی هسته شبیه‌ساز (Simulation Core)**

در این فاز، ما چارچوب اصلی محیط شبیه‌سازی را می‌سازیم.

* **۱.۱. تعریف محیط (Environment):**
* **کلاس GridWorld یا GraphWorld:** یک کلاس برای نمایش دنیای شبیه‌سازی. این می‌تواند یک شبکه توری ساده (Grid) یا یک گراف پیچیده‌تر (نمایشگر تقاطع‌ها به عنوان گره و جاده‌ها به عنوان یال) باشد.
* **ویژگی‌ها:** ابعاد، موانع، ویژگی‌های هر منطقه (مانند محدودیت سرعت، وجود ایستگاه شارژ).
* **۱.۲. مدل‌سازی موجودیت‌های متحرک (Mobile Agents):**
* **کلاس Agent:** یک کلاس پایه انتزاعی برای تمام موجودیت‌های متحرک.
* **ویژگی‌ها:** id, position, velocity, destination, battery\_level.
* **متدها:** move(), update\_state(), plan\_route().
* **کلاس‌های فرزند:** VehicleAgent, DroneAgent, DataPacketAgent که از Agent ارث‌بری می‌کنند و ویژگی‌های خاص خود را دارند.
* **۱.۳. پیاده‌سازی موتور زمان (Simulation Engine):**
* **یک حلقه اصلی (Main Loop):** این حلقه، زمان شبیه‌سازی را به صورت گسسته (Tick-based) جلو می‌برد.
* **در هر تیک (Tick):**

1. وضعیت تمام Agentها را به‌روزرسانی کن (agent.update\_state()).
2. به Agentها اجازه حرکت بده (agent.move()).
3. رویدادهای جدید (مانند رسیدن به مقصد) را پردازش کن.
4. داده‌های شبیه‌سازی را برای تحلیل و نمایش ثبت کن.

**ابزارهای پیشنهادی برای این فاز:**

* **زبان:** Python
* **کتابخانه‌ها:**
* NumPy: برای محاسبات عددی سریع (موقعیت، سرعت).
* NetworkX: برای مدل‌سازی محیط به صورت گراف (بسیار قدرتمند برای مسیریابی).
* Matplotlib / Seaborn: برای نمایش‌های اولیه و استاتیک از وضعیت محیط.

**فاز ۲: ادغام مؤلفه دانش (Knowledge-aware Component)**

حالا هوش را به سیستم اضافه می‌کنیم.

* **۲.۱. ساخت پایگاه دانش (Knowledge Base):**
* **کلاس KnowledgeBase:** یک کلاس متمرکز برای مدیریت دانش.
* **دانش ایستا:** بارگذاری نقشه‌ها (از NetworkX graph)، قوانین و محدودیت‌ها از فایل‌های کانفیگ (مانند JSON یا YAML).
* **دانش پویا:** این کلاس باید متدهایی برای دریافت و ذخیره‌سازی داده‌های زنده (شبیه‌سازی شده) داشته باشد. مثلاً update\_traffic(road\_id, density) یا log\_event(event\_type, position).
* **۲.۲. مدل‌سازی ماژول تصمیم‌گیری (Decision-making Module):**
* **کلاس DecisionMaker:** این مغز سیستم است.
* **متدها:**
* find\_optimal\_route(agent, destination): این متد از KnowledgeBase برای پیدا کردن مسیر بهینه استفاده می‌کند (مثلاً با الگوریتم A\* که وزن یال‌ها (جاده‌ها) به صورت پویا بر اساس ترافیک فعلی از KnowledgeBase خوانده می‌شود).
* assign\_resource(agent, resource\_type): مثلاً نزدیک‌ترین ایستگاه شارژ خالی را پیدا می‌کند.
* **۲.۳. ادغام با Agentها:**
* Agentها دیگر به صورت تصادفی یا ساده حرکت نمی‌کنند. قبل از حرکت، آن‌ها از DecisionMaker یک برنامه (Plan) درخواست می‌کنند.

**فاز ۳: پیاده‌سازی مدل‌های یادگیری ماشین (Machine Learning Integration)**

این فاز SkyMind را به یک سیستم واقعاً هوشمند و پیش‌بین تبدیل می‌کند.

* **۳.۱. مدل پیش‌بینی ترافیک (Traffic Prediction):**
* **داده‌ها:** از داده‌های تاریخی ثبت شده در KnowledgeBase (ترافیک در زمان‌های مختلف) استفاده کنید.
* **مدل:** یک مدل سری زمانی مانند LSTM یا GRU (با استفاده از PyTorch یا TensorFlow) برای پیش‌بینی تراکم ترافیک در آینده نزدیک.
* **ادغام:** DecisionMaker حالا می‌تواند برای مسیریابی، به جای ترافیک *فعلی*، از ترافیک *پیش‌بینی شده* برای ۳۰ دقیقه آینده استفاده کند.
* **۳.۲. مدل پیش‌بینی تقاضا (Demand Prediction):**
* **داده‌ها:** تاریخچه درخواست‌ها برای سرویس (مثلاً درخواست تاکسی).
* **مدل:** یک مدل رگرسیون یا دسته‌بندی برای پیش‌بینی مناطقی که در آینده نزدیک تقاضای بالایی خواهند داشت.
* **ادغام:** سیستم می‌تواند Agentها (خودروهای خالی) را به صورت پیش‌دستانه (Proactively) به سمت مناطق پرتقاضا هدایت کند تا زمان انتظار مشتری کاهش یابد.

**ابزارهای پیشنهادی برای این فاز:**

* **کتابخانه‌ها:**
* PyTorch / TensorFlow: برای ساخت و آموزش مدل‌های یادگیری عمیق.
* Scikit-learn: برای مدل‌های کلاسیک‌تر و پیش‌پردازش داده‌ها.

**فاز ۴: توسعه داشبورد تعاملی (Interactive Dashboard)**

این فاز، نتایج شبیه‌سازی را زنده و قابل فهم می‌کند.

* **۴.۱. انتخاب تکنولوژی:**
* **راه‌حل پیشنهادی:** Dash by Plotly. این یک چارچوب پایتونی است که به شما اجازه می‌دهد با استفاده از پایتون خالص، وب اپلیکیشن‌های تحلیلی و داشبوردهای تعاملی بسازید. این چارچوب بر روی Flask, Plotly.js و React.js ساخته شده است.
* **جایگزین‌ها:** Streamlit, Panel, Bokeh.
* **۴.۲. طراحی اجزای داشبورد:**
* **نمای اصلی (Main View):** یک نقشه تعاملی که موقعیت لحظه‌ای Agentها را نشان می‌دهد. رنگ Agentها می‌تواند وضعیت آن‌ها را نشان دهد (مثلاً سبز=در حال حرکت، آبی=در حال شارژ، قرمز=منتظر). رنگ جاده‌ها می‌تواند تراکم ترافیک را نشان دهد.
* **کنترل‌های شبیه‌سازی (Simulation Controls):** دکمه‌های Play/Pause/Reset، اسلایدر برای کنترل سرعت شبیه‌سازی، فیلدهایی برای تغییر پارامترها (مانند تعداد Agentها).
* **نمودارهای آماری (Statistical Charts):**
* نمودار خطی برای نمایش معیارهای کلیدی در طول زمان (میانگین زمان سفر، تعداد کل ماموریت‌های انجام شده، سطح باتری میانگین).
* هیستوگرام برای توزیع سرعت‌ها یا زمان‌های انتظار.
* **نمای جزئیات (Drill-down View):** با کلیک بر روی یک Agent یا یک جاده، اطلاعات دقیق‌تری در یک پنل کناری نمایش داده شود (ID ایجنت، مقصد، مسیر برنامه‌ریزی شده، ترافیک فعلی جاده).
* **۴.۳. اتصال داشبورد به شبیه‌ساز:**
* شبیه‌ساز باید در هر تیک، وضعیت کلی سیستم را در یک ساختار داده مشخص (مثلاً یک فایل JSON یا یک پایگاه داده in-memory مانند Redis) به‌روزرسانی کند.
* اپلیکیشن Dash به صورت دوره‌ای (مثلاً هر ثانیه) این داده‌ها را می‌خواند و تمام اجزای بصری خود را با استفاده از Callbacks به‌روزرسانی می‌کند.