PROVISIONAL PATENT APPLICATION

Protocol-as-Kernel Architecture for AI Operating Systems with Zero Static Configuration

Inventor: Jean-Patrick Smith

Filing Date: 5/30/2025

Application Type: Provisional Patent Application

ABSTRACT

This invention discloses a protocol-as-kernel architecture for AI operating systems that eliminates static configuration dependencies through dynamic context assembly and intent-driven resource allocation. Unlike traditional AI orchestration systems that rely on predefined prompt templates and manual configuration, the disclosed system implements protocols as the fundamental kernel primitive, enabling zero static prompt operation through real-time context construction. The architecture features a protocol dispatcher that routes messages based on semantic intent analysis, a context assembly system that dynamically constructs operational contexts without static templates, and cross-context learning mechanisms that optimize performance through federated knowledge sharing. Key innovations include protocol-native hardware acceleration, intent-driven resource allocation, and privacy-preserving cross-context optimization that collectively enable unprecedented scalability and adaptability in AI system architectures.

TECHNICAL FIELD

This invention relates to artificial intelligence operating systems, distributed computing architectures, and protocol-based system design. More specifically, the invention concerns novel kernel architectures that treat protocols as fundamental system primitives rather than application-layer abstractions, enabling dynamic context assembly and intent-driven resource allocation without static configuration dependencies.

The technical field encompasses several converging domains: AI orchestration systems that coordinate multiple AI models and services; protocol design methodologies for distributed systems; kernel architecture innovations for modern computational workloads; context management systems for large language models and AI applications; and resource allocation algorithms for heterogeneous computational environments.

BACKGROUND

Current AI operating systems and orchestration platforms exhibit fundamental limitations in scalability, adaptability, and configuration complexity. These limitations stem from architectural decisions that treat protocols as application-layer concerns rather than kernel-level primitives, necessitating extensive static configuration and manual optimization.

Existing AI Operating System Limitations:

Traditional AI orchestration systems like LangChain, AutoGen, and CrewAI implement agent-based architectures that require extensive manual configuration for each operational scenario. These systems rely on static prompt templates, predefined agent roles, and manual workflow specification that creates significant scalability barriers and adaptation overhead.

Static Configuration Dependencies:

Current systems require developers to manually define prompt templates, specify agent interactions, configure resource allocation policies, and establish communication protocols before deployment. This static configuration approach creates brittleness when operational requirements change and necessitates manual intervention for optimization.

Protocol Layer Limitations:

While the Model Context Protocol (MCP) provides standardized interfaces for tool integration, current implementations treat MCP as an application-layer integration mechanism rather than a fundamental kernel primitive. This limits the protocol's potential for enabling dynamic system optimization and automated resource allocation.

Resource Allocation Inefficiencies:

Existing systems typically implement naive resource allocation strategies that treat all computational tasks as equivalent consumers, failing to leverage semantic understanding of computational requirements for optimization. This results in suboptimal hardware utilization and unnecessary resource waste.

Context Management Challenges:

Current large language model applications rely on static prompt engineering and manual context management, creating maintenance overhead and limiting adaptability to varying operational conditions. The dependency on predefined context structures prevents dynamic optimization based on actual usage patterns.

DETAILED DESCRIPTION

Core System Architecture

The disclosed protocol-as-kernel architecture fundamentally reimagines AI operating system design by treating protocols as the primary kernel abstraction rather than traditional process and memory management primitives. This architectural innovation enables dynamic resource allocation, automated context assembly, and intelligent optimization without static configuration dependencies.

Protocol Dispatcher (101): The protocol dispatcher serves as the central kernel component that receives, analyzes, and routes protocol messages based on semantic content and intent analysis. Unlike traditional message routing that relies on static destination addresses, the protocol dispatcher performs real-time intent classification to determine optimal processing strategies.

The dispatcher implements a multi-protocol parser framework that automatically detects and validates incoming message formats without requiring predefined schema definitions. This enables the system to adapt to new protocol formats at runtime without manual configuration or system updates.

Intent Classification Engine (103): The intent classification engine performs multi-modal analysis of incoming requests to determine user intentions and optimal resource allocation strategies. The engine implements natural language processing algorithms, behavioral pattern recognition, and context-aware classification that enables accurate intent determination even for ambiguous or incomplete requests.

The classification process incorporates real-time system state information, historical interaction patterns, and available resource capabilities to ensure classified intentions are feasible within current operational constraints. This prevents over-allocation and enables graceful degradation when resources are constrained.

Context Assembly System (104): The context assembly system eliminates dependency on static prompt templates through dynamic context construction based on classified intentions and available information sources. The system implements semantic context decomposition algorithms that identify essential context elements without relying on predefined categorizations.

Context relevance scoring algorithms evaluate potential context elements using multi-dimensional analysis including semantic similarity, temporal relevance, source reliability, and computational cost. This enables optimal context selection without manual tuning or static threshold configuration.

Resource Allocation Manager (108): The resource allocation manager coordinates computational resources across heterogeneous hardware environments through semantic-aware allocation algorithms. Unlike traditional resource managers that treat all tasks equivalently, the disclosed system incorporates intent classification results and computational requirement analysis to optimize resource utilization.

The manager implements predictive resource allocation that anticipates future requirements based on current operational patterns and historical usage analysis. This enables proactive resource provisioning and prevents resource contention bottlenecks.

Cross-Context Learning Module (110): The cross-context learning module enables knowledge accumulation and sharing across different operational contexts through privacy-preserving federated learning algorithms. The module implements secure knowledge exchange protocols that enable system-wide optimization without compromising sensitive information.

Learning propagation mechanisms ensure beneficial optimizations spread throughout the system while knowledge conflict resolution processes handle potentially contradictory optimization strategies from different contexts.

Zero Static Prompt Operation

The disclosed system achieves zero static prompt operation through dynamic context assembly algorithms that construct operational contexts at runtime based on current requirements and available information sources. This eliminates the need for predefined prompt templates and enables automatic adaptation to varying operational conditions.

Dynamic Context Assembly Algorithms: The context assembly process implements semantic decomposition of user requests into fundamental components including explicit objectives, implicit requirements, contextual constraints, and performance preferences. This decomposition enables understanding of essential requirements without relying on predefined categorizations or static templates.

Context relevance scoring algorithms evaluate potential context elements using multi-dimensional analysis that considers semantic similarity to current objectives, temporal relevance of information sources, reliability and accuracy of data sources, and computational cost of including specific context elements.

Template-Free Prompt Generation: The system generates prompts dynamically by combining relevant context elements according to learned patterns of effective organization. The generation process incorporates feedback from operational outcomes to continuously improve prompt construction strategies without manual template maintenance.

Cross-context prompt sharing mechanisms enable beneficial prompt construction patterns to propagate across different operational domains while maintaining privacy and security constraints through differential privacy algorithms and secure multi-party computation protocols.

Intent-Driven Context Construction: Context construction algorithms incorporate intent classification results to prioritize relevant information sources and exclude potentially distracting or conflicting context elements. This ensures constructed contexts remain focused on current objectives while providing comprehensive situational awareness.

The system implements adaptive context window management that dynamically adjusts context size based on computational constraints and quality requirements, enabling optimal trade-offs between context comprehensiveness and processing efficiency.

Implementation Details

Protocol Parsing Methods: The system implements a multi-protocol parser framework that performs automatic format detection through statistical analysis, pattern recognition, and schema inference algorithms. This enables support for diverse protocol formats without requiring manual parser configuration or predefined schema definitions.

Parser adaptation mechanisms enable automatic updates to parsing logic when new protocol variants are encountered, ensuring robust operation across evolving protocol ecosystems without manual maintenance overhead.

Kernel Interface Specifications: The kernel provides protocol-native APIs that enable direct protocol manipulation and optimization at the system level. These interfaces allow applications to interact with the protocol dispatcher through standardized methods while enabling system-level optimizations based on semantic protocol analysis.

Hardware acceleration interfaces enable direct integration with protocol processing units, FPGA-based acceleration systems, and specialized AI hardware through abstracted interfaces that maintain performance characteristics across different hardware configurations.

Cross-Context Learning Mechanisms: The system implements federated learning algorithms that enable knowledge sharing across operational contexts without requiring centralized data aggregation. Learning algorithms incorporate differential privacy mechanisms to ensure sensitive information remains protected while enabling beneficial optimization sharing.

Knowledge validation mechanisms ensure shared optimizations maintain accuracy and effectiveness across different operational contexts through automated testing, performance verification, and rollback capabilities when optimizations prove ineffective.

DESCRIPTION OF DRAWINGS

FIGURE 1 - Overall System Architecture (101-112): Illustrates the complete protocol-as-kernel architecture showing the protocol dispatcher (101) at the core, accepting inputs from user applications and external services through the protocol input interface (102). The intent classification engine (103) processes incoming requests while the context assembly system (104) dynamically constructs operational contexts. Hardware resources (109a-109d) including CPU clusters, GPU arrays, AI accelerators, and protocol processors interface through the hardware interface layer (112).

FIGURE 2 - Protocol Dispatcher Details (201-214): Shows the internal architecture of the protocol dispatcher including the multi-protocol parser (201), dynamic protocol schema definitions (202), semantic routing engine (203), and intent classification pipeline (205). The intent-to-resource mapping algorithms (209) translate classified intents to resource allocation pathways (210), with decision points and flow control (214) enabling adaptive optimization.

FIGURE 3 - Context Assembly Process (301-313): Depicts the dynamic context assembly workflow from semantic decomposition (301) of user requests through context relevance scoring (302), dynamic assembly logic (303), and memory integration methods (305). Context inheritance patterns (309) enable efficient knowledge propagation while context filtering and sanitization (313) ensures quality and relevance.

FIGURE 4 - Zero Static Prompt Operation Comparison (401-419): Contrasts traditional static prompt template architectures (401-403) with the Kingly OS zero static prompt operation (411-419), highlighting how intent classification (411), dynamic context assembly (413), and runtime prompt generation (415) eliminate the need for predefined templates while providing superior contextual awareness.

FIGURE 5 - Intent Classification System Architecture (501-517): Details the multi-modal intent classification engine showing input layer (502) processing of natural language, voice, visual, and contextual history inputs. The semantic analysis layer (506), intent classification core (508), and contextual refinement (510) components work together to produce classified intents with confidence scores and resource requirements.

FIGURE 6 - Hardware Integration Architecture (601-616): Illustrates the hardware stack from application layer (602) through protocol-as-kernel layer (604), hardware abstraction layer (606), to physical hardware (608). Specialized acceleration components (610) including FPGA protocol accelerators, custom ASIC chips, and protocol processing units enable protocol-native hardware optimization.

FIGURE 7 - Cross-Context Learning Flow (701-721): Shows the cross-context learning module architecture with experience aggregation (704), knowledge synthesis (706), privacy-preserving storage (708), and adaptive optimization (710). Learning feedback loops (712) enable continuous improvement across protocol efficiency, intent classification accuracy, and resource management performance.

DRAWINGS

FIGURE 1 - OVERALL SYSTEM ARCHITECTURE

```
FIGURE 1 - OVERALL SYSTEM ARCHITECTURE

┌───────────────────────────────────────────────────────────────────────┐
│                 KINGLY OS PROTOCOL-AS-KERNEL ARCHITECTURE             │
└───────────────────────────────────────────────────────────────────────┘

External Protocol Sources                    System Core
Hardware Resources
┌──────────────────────┐                       ┌──────────────────────┐
┌──────────────────────┐
│ User Applications    │◄─────────────────────►│   PROTOCOL           │◄──────────►│
CPU Cluster (109a)   │
│ (MCP, HTTP, WS)      │                       │   DISPATCHER         │            │
│
└──────────────────────┘                       │   (101) [KERNEL]     │
└──────────────────────┘
                                               └──────────────────────┘
                                                         │
┌──────────────────────┐
┌──────────────────────┐                                 │                        │
GPU Array (109b)     │
│ External Services    │                                 ▼                        │
│
│ (APIs, Databases)    │──► Protocol Input ──►  ┌──────────────────┐
└──────────────────────┘
└──────────────────────┘    Interface (102)     │ INTENT CLASS.    │
┌──────────────────────┐
                                                │ ENGINE (103)     │        │
AI Accelerators      │
┌──────────────────────┐                        └──────────────────┘        │
(TPU/NPU) (109c)     │                                   │
│ Real-Time Data       │                                 │
└──────────────────────┘
│ Streams (106)        │─────────────────────────────────┼──────────────────────►
┌──────────────────────┐
└──────────────────────┘                                 ▼                  │
Protocol Processors  │
                                                                            │
(FPGA/ASIC) (109d)   │                          ┌──────────────────┐
Memory Systems
└──────────────────────┘                          │ CONTEXT          │

┌──────────────────────┐                          │ ASSEMBLY         │
│ Long-Term Memory     │◄────────────────────────►│ SYSTEM (104)     │
│ (105)                │                          └──────────────────┘
└──────────────────────┘                                    │
                                                            ▼
External Interfaces
Learning Network                                  ┌──────────────────┐
┌──────────────────────┐                          │ RESOURCE         │
┌──────────────────┐
│ Protocol Gateway     │◄────────────────────────►│ ALLOCATION       │
│◄──────────►│ Knowledge           │
│ (107)                │                          │ MANAGER (108)    │
│ Sharing (111)    │
```

```
└──────────────────────┘                        └─────────┬─────────┘
└────────────────────┘                                    │

                                                          │
▲

│                                               ┌─────────▼─────────┐

│◄───────────────────┘                          │ HARDWARE

                                                │ INTERFACE         │
                                                │ LAYER (112)       │
                                                └───────────────────┘
                                                          │
                                                ┌─────────▼─────────┐
                                                │ CROSS-CONTEXT     │
                                                │ LEARNING          │
                                                │ MODULE (110)      │
                                                └───────────────────┘

KEY ARCHITECTURAL PRINCIPLES:
├─ Protocol-First Abstraction: All operations via protocol exchanges
├─ Intent-Driven Operation: Semantic understanding drives resource
allocation
├─ Dynamic Assembly: Runtime configuration without static templates
├─ Context-Aware Management: Resource decisions based on semantic
understanding
└─ Universal Scaling: Same architecture scales from single-user to
distributed

INNOVATION HIGHLIGHTS:
• Protocol Dispatcher (101) replaces traditional OS kernel abstractions
• Zero static configuration through Intent Classification Engine (103)
• Dynamic Context Assembly (104) eliminates prompt templates
• Cross-Context Learning (110) enables knowledge accumulation
• Hardware Interface Layer (112) provides protocol-based hardware
abstraction
```

FIGURE 2 - PROTOCOL DISPATCHER DETAILS

```
┌──────────────────────────────────────────────────────────────────────────
┐
│                      PROTOCOL DISPATCHER ARCHITECTURE
│
└──────────────────────────────────────────────────────────────────────────
┘

Input Protocol Messages                        Dispatcher Core
Output Pathways

                                              ┌──────────────────────────┐
┌─────────────────┐                           │   MULTI-PROTOCOL         │
│ MCP Messages    │──────────────────────────►│   PARSER (201)           │
└─────────────────┘                           │                          │
```

```
                              │ ┌──────────────────────┐ │
┌──────────────────┐          │ │                      │ │
│ HTTP/HTTPS       │─────────►│ │ Dynamic Protocol     │ │
└──────────────────┘          │ │ Schema Defs (202)    │ │
┌──────────────────┐          │ │                      │ │
┌──────────────────┐          │ └──────────────────────┘ │◄────────►│
Resource Pool A │
│ WebSocket        │─────────►│                          │          │
(CPU Intensive) │
└──────────────────┘          └──────────┬───────────────┘
└──────────────────┘                     │
┌──────────────────┐                     │
┌──────────────────┐                     │
│ gRPC             │─────────────────────┬▼───────────────────┐     │
Resource Pool B │
└──────────────────┘            │ SEMANTIC ROUTING     │◄──────►│
(GPU Compute)   │
┌──────────────────┐            │ ENGINE (203)         │
└──────────────────┘
│ Custom AI        │────────────┤                      │
┌──────────────────┐
│ Protocols        │            │  ┌──────────────────┐│        │
Resource Pool C │
└──────────────────┘            │  │Protocol Trans    ││◄──────►│
(AI Accelerator)│
                                │  │Layer (204)       ││
└──────────────────┘
                                │  └──────────────────┘│
┌──────────────────┐
                                └──────────┬───────────┘        │
Resource Pool D │
                                           │                    │
(Memory/Storage)│
                                           ▼
└──────────────────┘

                          ┌──────────────────────────────────┐
                          │    INTENT CLASSIFICATION         │
                          │      PIPELINE (205)              │
                          │                                  │
                          │  ┌─────────────────────────────┐ │
                          │  │ NLP Modules (206)           │ │
                          │  │ ├─ Context Understanding    │ │
                          │  │ ├─ Domain Adaptation        │ │
                          │  │ └─ Intent Disambiguation    │ │
                          │  └─────────────────────────────┘ │
                          │                                  │
                          │  ┌─────────────────────────────┐ │
                          │  │ Multi-Modal Recognition     │ │
                          │  │ (207)                       │ │
                          │  │ ├─ Text Processing          │ │
                          │  │ ├─ Structured Data          │ │
                          │  │ ├─ Voice Input              │ │
                          │  │ └─ Behavioral Patterns      │ │
                          │  └─────────────────────────────┘ │
                          │                                  │
```

```
                             │ ┌──────────────────────────────┐ │
                             │ │ Context-Aware Classification │ │
                             │ │ (208)                        │ │
                             │ │ ├─ System State (211)        │ │
                             │ │ ├─ Hardware Caps (212)       │ │
                             │ │ └─ QoS Requirements (213)    │ │
                             │ └──────────────────────────────┘ │
                             └─────────────┬────────────────────┘
                                           │
                                           ▼
                             ┌──────────────────────────────────┐
                             │  INTENT-TO-RESOURCE MAPPING      │
                             │       ALGORITHMS (209)           │
                             │                                  │
                             │ ┌──────────────────────────────┐ │
                             │ │ Dynamic Resource Assessment  │ │
                             │ │ Optimization Strategy Select │ │
                             │ │ Load Balancing Integration   │ │
                             │ │ QoS Optimization             │ │
                             │ └──────────────────────────────┘ │
                             └─────────────┬────────────────────┘
                                           │
                                           ▼
                             ┌──────────────────────────────────┐
                             │   RESOURCE ALLOCATION            │
                             │    PATHWAYS (210)                │
                             │                                  │
                             │ ┌─ Pathway A: High Priority ─────┤
                             │ ├─ Pathway B: Standard Proc ─────┤
                             │ ├─ Pathway C: Background Task ───┤
                             │ └─ Pathway D: Emergency Route ───┤
                             └──────────────────────────────────┘
                                           │
                                           ▼
                             ┌──────────────────────────────────┐
                             │  DECISION POINTS & FLOW          │
                             │    CONTROL (214)                 │
                             │                                  │
                             │ ┌─ Execution Path Selection      │
                             │ ├─ Performance Optimization      │
                             │ ├─ Failure Recovery Routes       │
                             │ └─ Dynamic Reconfiguration         │
                             └──────────────────────────────────┘

PROCESSING FLOW:
1. Multi-Protocol Parser (201) receives and validates incoming messages
2. Dynamic Protocol Schema Definitions (202) enable runtime protocol
support
3. Semantic Routing Engine (203) performs intent-aware message dispatching
4. Intent Classification Pipeline (205) determines user objectives and
requirements
5. Intent-to-Resource Mapping (209) translates intentions to optimal
strategies
6. Resource Allocation Pathways (210) direct requests to appropriate
```

```
resources
7. Decision Points (214) enable adaptive optimization and error recovery

KEY INNOVATIONS:
• Kernel-level protocol processing eliminates application-layer overhead
• Intent-aware routing considers semantic content, not just destination
• Dynamic protocol schema support enables runtime protocol evolution
• Multi-modal intent recognition supports diverse communication channels
• Context-aware classification ensures feasible resource allocation
```

FIGURE 3 - CONTEXT ASSEMBLY PROCESS

```
┌──────────────────────────────────────────────────────────────────────────
┐
│                    DYNAMIC CONTEXT ASSEMBLY WORKFLOW
│
└──────────────────────────────────────────────────────────────────────────
┘

Input Request                    Assembly Engine
Context Sources
┌─────────────────┐         ┌─────────────────────────────┐
┌─────────────────┐
│ User Request    │────────►│   SEMANTIC DECOMPOSITION    │      │ Long-
Term        │
│ "Analyze data   │         │        (301)                │◄────►│ Memory
(306)    │
│ and create      │         │                             │      │ ├─
Domain Know  │
│ visualization"  │         │  ┌─────────────────────────┐ │      │ ├─ User
Prefs    │
└─────────────────┘         │ │ Explicit Objectives:    │ │      │ └─ Best
Practice│
                            │ │ • Data Analysis         │ │
└─────────────────┘
┌─────────────────┐         │ │ • Visualization Creation│ │
┌──────────────────┐
│ System Context  │────────►│ │                         │ │      │ Short-
Term        │
│ • Available     │         │ │ Implicit Requirements:  │ │◄────►│ Context
(307)    │
│   Tools         │         │ │ • Data Format Support   │ │      │ ├─
Session State │
│ • Current Load  │         │ │ • Visual Aesthetics     │ │      │ ├─
Recent Actions│
│ • Resources     │         │ │ • Performance Needs     │ │      │ └─
Active Tasks  │
└─────────────────┘         │ └─────────────────────────┘ │
└──────────────────┘
                            └──────────────┬──────────────┘

┌──────────────────┐
```

```
                                              │                      │ Real-
Time        │
                                              ▼                      │ Data
(308)       │
                                 ┌──────────────────────────────┐    │ ├
System Metrics│
                                 │   CONTEXT RELEVANCE          │◄──►│ ├
External APIs │
                                 │   SCORING SYSTEM (302)       │    │ └ Live
Feeds    │
                                 │                              │
 └───────────────────┘
                                 │ Scoring Dimensions:          │
                                 │ ├─ Semantic Similarity       │
                                 │ ├─ Temporal Relevance        │
                                 │ ├─ Source Reliability        │
                                 │ ├─ Computational Cost        │
                                 │ └─ User Preference Weight    │
                                 └──────────────────────────────┘
                                              │
                                              ▼
                                 ┌───────────────────────────────┐
                                 │    DYNAMIC ASSEMBLY           │
                                 │      LOGIC (303)              │
                                 │                               │
                                 │  ┌───────────────────────────┐│
                                 │  │ Context Element Selection ││
                                 │  │ ├─ High Relevance Items   ││
                                 │  │ ├─ Essential Context      ││
                                 │  │ ├─ Performance Balance    ││
                                 │  │ └─ Coherence Optimization ││
                                 │  └───────────────────────────┘│
                                 │                               │
                                 │  ┌───────────────────────────┐│◄── Operational
                                 │  │ Feedback Integration      ││    Outcomes
(304)
                                 │  │ ├─ Success Patterns       ││
                                 │  │ ├─ Failure Analysis       ││
                                 │  │ └─ Strategy Adaptation    ││
                                 │  └───────────────────────────┘│
                                 └───────────────────────────────┘
                                              │
                                              ▼
                                 ┌──────────────────────────────┐
                                 │    MEMORY INTEGRATION        │
                                 │     METHODS (305)            │
                                 │                              │
                                 │  ┌─ Long-Term Memory (306) ──│
                                 │  │ ├─ Domain Knowledge Base  │
                                 │  │ ├─ User Behavior Patterns │
                                 │  │ └─ Optimization History   │
                                 │  │                           │
                                 │  ├─ Short-Term Context (307)─│
                                 │  │ ├─ Session Variables      │
│ │ ├─ Recent Interactions      │
│ │ └─ Active State Data        │
│ │                             │
│ └─ Real-Time Streams (308)────│
│   ├─ Live System Metrics      │
│   ├─ External Data Feeds      │
│   └─ Dynamic Constraints      │
└─────────────┬─────────────────┘
              │
              ▼
┌─────────────────────────────┐
│   CONTEXT INHERITANCE       │
│     PATTERNS (309)          │
│                             │
│ ┌─ Hierarchical Structures ─│
│ │ (310)                     │
│ │ ├─ Parent-Child Relations │
│ │ ├─ Sibling Context Sharing│
│ │ └─ Selective Inheritance  │
│ │                           │
│ ├─ Temporal Evolution (311)─│
│ │ ├─ Context Aging          │
│ │ ├─ Relevance Decay        │
│ │ └─ Update Propagation     │
│ │                           │
│ └─ Cross-Domain Mapping ────│
│   (312)                     │
│   ├─ Domain Translation     │
│   ├─ Semantic Bridging      │
│   └─ Knowledge Transfer     │
└─────────────┬───────────────┘
              │
              ▼
┌─────────────────────────────┐
│   CONTEXT FILTERING &       │
│   SANITIZATION (313)        │
│                             │
│ ├─ Relevance Filtering      │
│ ├─ Conflict Resolution        │
│ ├─ Information Validation   │
│ ├─ Privacy Protection       │
│ └─ Quality Assurance        │
└─────────────┬───────────────┘
              │
              ▼
┌─────────────────────────────┐
│    ASSEMBLED CONTEXT        │
│        OUTPUT               │
│                             │
│ ┌─ Coherent Context Package─│
│ ├─ Optimized for Performance│
│ ├─ Semantically Rich        │
│ ├─ Dynamically Constructed  │
│ └─ Zero Static Templates    │
```

```
                              └──────────────────────────────┘

ASSEMBLY PRINCIPLES:
• Semantic Decomposition (301): Break requests into fundamental components
• Relevance Scoring (302): Multi-dimensional evaluation of context
elements
• Dynamic Assembly (303): Runtime construction using learned patterns
• Memory Integration (305): Combine multiple information sources
coherently
• Context Inheritance (309): Efficient propagation across related
operations
• Quality Assurance (313): Ensure relevance and accuracy throughout
process

INNOVATION HIGHLIGHTS:
✓ Eliminates static prompt templates through dynamic construction
✓ Multi-source memory integration for comprehensive situational awareness
✓ Context inheritance enables efficient knowledge sharing
✓ Real-time adaptation based on operational feedback
✓ Semantic coherence maintained throughout assembly process
```

FIGURE 4 - ZERO STATIC PROMPT OPERATION COMPARISON

```
Traditional Static Prompt Architecture:
┌──────────────────────────────────────────────────────────────────────────
┐
│                            TRADITIONAL AI SYSTEM
│
├──────────────────────────────────────────────────────────────────────────
┤
│  User Intent: "Schedule meeting with team about project"
│
│                                 ↓
│
│  ┌──────────────────────────────────────────────────────────────────┐
│
│  │               STATIC PROMPT TEMPLATES (401)                      │
│
│  │  ┌─────────────┐  ┌─────────────┐  ┌─────────────┐                │
│
│  │  │ Calendar    │  │ Email       │  │ Meeting     │                │
│
│  │  │ Template    │  │ Template    │  │ Template    │                │
│
│  │  │ (402a)      │  │ (402b)      │  │ (402c)      │                │
│
│  │  └─────────────┘  └─────────────┘  └─────────────┘                │
│
│  └──────────────────────────────────────────────────────────────────┘
│
│                                 ↓
```

```
│
│  ┌──────────────────────────────────────────────────────────┐
│  │              TEMPLATE SELECTOR (403)                     │
│  │        [Limited to predefined templates]                  │
│  └──────────────────────────────────────────────────────────┘
│                              ↓
│  Output: Generic meeting scheduling with limited context
└─────────────────────────────────────────────────────────────────
┘
                                VS

Kingly OS Zero Static Prompt Operation:
┌─────────────────────────────────────────────────────────────────
┐
│                        KINGLY OS SYSTEM
│
├─────────────────────────────────────────────────────────────────
┤
│  User Intent: "Schedule meeting with team about project"
│                              ↓
│  ┌──────────────────────────────────────────────────────────┐
│  │              INTENT CLASSIFICATION ENGINE (411)          │
│  │  ┌────────────┐ ┌────────────┐ ┌────────────┐             │
│  │  │ Semantic   │ │ Context    │ │ Multi-modal │             │
│  │  │ Analysis   │ │ History    │ │ Recognition │             │
│  │  │ (412a)     │ │ (412b)     │ │ (412c)      │             │
│  │  └────────────┘ └────────────┘ └────────────┘             │
│  └──────────────────────────────────────────────────────────┘
│                              ↓
│  ┌──────────────────────────────────────────────────────────┐
│  │          DYNAMIC CONTEXT ASSEMBLY SYSTEM (413)           │
│  │  ┌────────────┐ ┌────────────┐ ┌────────────┐             │
│  │  │ Team       │ │ Project    │ │ Calendar    │             │
│
```

```
│  │  │ Profiles     │ │ Context     │ │ Availability│              │
│  │  │ (414a)      │ │ (414b)      │ │ (414c)      │              │
│  │  └─────────────┘ └─────────────┘ └─────────────┘              │
│  │                            │                                  │
│  │  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐              │
│  │  │ Meeting     │ │ Time Zone   │ │ Resource    │              │
│  │  │ History     │ │ Data        │ │ Allocation  │              │
│  │  │ (414d)      │ │ (414e)      │ │ (414f)      │              │
│  │  └─────────────┘ └─────────────┘ └─────────────┘              │
│  └───────────────────────────────────────────────────────────────┘
│                              ↓
│  ┌───────────────────────────────────────────────────────────────┐
│  │         RUNTIME PROMPT GENERATION (415)                       │
│  │         [Contextually optimized for this specific intent]      │
│  └───────────────────────────────────────────────────────────────┘
│                              ↓
│  Output: Intelligent meeting scheduling with full contextual awareness
│          and cross-protocol resource coordination
│
└───────────────────────────────────────────────────────────────────────────
┘

Key Innovation Elements:
(416) Zero Template Storage - No static prompts stored in system
(417) Runtime Context Assembly - Dynamic composition from live data
sources
(418) Cross-Protocol Intelligence - Unified context across multiple
protocols
(419) Adaptive Learning - System improves from each interaction
```

FIGURE 5 - INTENT CLASSIFICATION SYSTEM ARCHITECTURE

```
┌───────────────────────────────────────────────────────────────────────────
┐
```

```
│                          INTENT CLASSIFICATION ENGINE
│
│                                     (501)
│
├──────────────────────────────────────────────────────────────────────
┤
│
│  INPUT LAYER (502)
│
│  ┌────────────┐ ┌────────────┐ ┌────────────┐ ┌────────────┐
│  │  Natural   │ │   Voice    │ │   Visual   │ │  Context   │
│  │  Language  │ │   Audio    │ │   Input    │ │  History   │
│  │   (503a)   │ │   (503b)   │ │   (503c)   │ │   (503d)   │
│  └────────────┘ └────────────┘ └────────────┘ └────────────┘
│        │             │             │             │
│        ↓             ↓             ↓             ↓
│  ┌──────────────────────────────────────────────────────────┐
│  │              MULTI-MODAL PREPROCESSING (504)             │
│  │  ┌────────────┐ ┌────────────┐ ┌────────────┐           │
│  │  │   Token    │ │   Audio    │ │   Image    │           │
│  │  │ Embedding  │ │ Feature    │ │ Feature    │           │
│  │  │   (505a)   │ │ Extraction │ │ Extraction │           │
│  │  │            │ │   (505b)   │ │   (505c)   │           │
│  │  └────────────┘ └────────────┘ └────────────┘           │
│  └──────────────────────────────────────────────────────────┘
│                              ↓
│  ┌──────────────────────────────────────────────────────────┐
│  │               SEMANTIC ANALYSIS LAYER (506)              │
│  │  ┌────────────┐ ┌────────────┐ ┌────────────┐           │
│  │  │   Entity   │ │Relationship│ │   Intent   │           │
│  │  │ Recognition│ │ Extraction │ │  Patterns  │           │
│
```

```
│  │  │   (507a)    │ │   (507b)    │ │   (507c)    │                 │
│  │  └─────────────┘ └─────────────┘ └─────────────┘                 │
│  └──────────────────────────────────────────────────────────────────┘
│                              ↓
│  ┌──────────────────────────────────────────────────────────────────┐
│  │            INTENT CLASSIFICATION CORE (508)                      │
│  │  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐                 │
│  │  │ Transformer │ │ Attention   │ │ Confidence   │                 │
│  │  │   Model     │ │ Mechanism   │ │ Scoring     │                 │
│  │  │   (509a)    │ │   (509b)    │ │   (509c)    │                 │
│  │  └─────────────┘ └─────────────┘ └─────────────┘                 │
│  └──────────────────────────────────────────────────────────────────┘
│                              ↓
│  ┌──────────────────────────────────────────────────────────────────┐
│  │              CONTEXTUAL REFINEMENT (510)                         │
│  │  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐                 │
│  │  │ Historical  │ │ User Model  │ │ Environment │                 │
│  │  │ Context     │ │ Preferences │ │ Context     │                 │
│  │  │   (511a)    │ │   (511b)    │ │   (511c)    │                 │
│  │  └─────────────┘ └─────────────┘ └─────────────┘                 │
│  └──────────────────────────────────────────────────────────────────┘
│                              ↓
│  ┌──────────────────────────────────────────────────────────────────┐
│  │                OUTPUT LAYER (512)                                │
│  │  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐                 │
│  │  │ Primary     │ │ Secondary   │ │ Required    │                 │
│  │  │ Intent      │ │ Intents     │ │ Resources   │                 │
│
```

```
│  │  │ (513a)       │ │ (513b)       │ │ (513c)       │                │
│  │  └──────────────┘ └──────────────┘ └──────────────┘                │
│  │                          │                                         │
│  │  ┌──────────────┐ ┌──────────────┐ ┌──────────────┐                │
│  │  │ Confidence   │ │ Alternative │ │ Context     │                │
│  │  │ Score        │ │ Options      │ │ Requirements│                │
│  │  │ (513d)       │ │ (513e)       │ │ (513f)       │                │
│  │  └──────────────┘ └──────────────┘ └──────────────┘                │
│  └──────────────────────────────────────────────────────────────────┘
│                              ↓
│ TO PROTOCOL DISPATCHER (101) & CONTEXT ASSEMBLY SYSTEM (104)
│
└──────────────────────────────────────────────────────────────────────
┘

Key Processing Flow:
(514) Multi-modal fusion enables understanding across input types
(515) Semantic analysis extracts deep meaning beyond surface keywords
(516) Contextual refinement personalizes classification to user patterns
(517) Confidence scoring enables graceful degradation and learning
```

FIGURE 6 - HARDWARE INTEGRATION ARCHITECTURE

```
┌──────────────────────────────────────────────────────────────────────
┐
│                      KINGLY OS HARDWARE STACK
│                              (601)
├──────────────────────────────────────────────────────────────────────
┤
│
│  APPLICATION LAYER (602)
│  ┌──────────────┐ ┌──────────────┐ ┌──────────────┐ ┌──────────────┐
│  │   AI Apps    │ │   System     │ │   Protocol   │ │    User      │
│  │   (603a)     │ │  Services    │ │  Handlers    │ │ Interface    │
│
```

```
│  │              │ │   (603b)     │ │   (603c)     │ │   (603d)     │
│  └──────────────┘ └──────────────┘ └──────────────┘ └──────────────┘
│
│                              ↕
│
│  ┌──────────────────────────────────────────────────────────────────┐
│  │              PROTOCOL-AS-KERNEL LAYER (604)                      │
│  │                                                                  │
│  │  ┌──────────────┐ ┌──────────────┐ ┌──────────────┐              │
│  │  │   Protocol   │ │    Intent    │ │   Context    │              │
│  │  │  Dispatcher  │ │Classificat.  │ │   Assembly   │              │
│  │  │    (101)     │ │    (103)     │ │    (104)     │              │
│  │  └──────────────┘ └──────────────┘ └──────────────┘              │
│  │                                │                                 │
│  │  ┌──────────────┐ ┌──────────────┐ ┌──────────────┐              │
│  │  │   Resource   │ │Cross-Context│ │   Hardware   │              │
│  │  │  Allocator   │ │   Learning   │ │ Accelerator  │              │
│  │  │    (105)     │ │    (110)     │ │  Interface   │              │
│  │  │              │ │              │ │    (605)     │              │
│  │  └──────────────┘ └──────────────┘ └──────────────┘              │
│  └──────────────────────────────────────────────────────────────────┘
│
│                              ↕
│
│  ┌──────────────────────────────────────────────────────────────────┐
│  │              HARDWARE ABSTRACTION LAYER (606)                    │
│  │                                                                  │
│  │  ┌──────────────┐ ┌──────────────┐ ┌──────────────┐              │
│  │  │     GPU      │ │    Neural    │ │    Memory    │              │
│  │  │  Scheduler   │ │  Processing  │ │  Controller  │              │
│  │  │    (607a)    │ │    Unit      │ │    (607c)    │              │
│
```

```
│  │  │             │ │   (607b)    │ │             │                │
│  │  └─────────────┘ └─────────────┘ └─────────────┘                │
│  │                               │                                 │
│  │  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐                │
│  │  │   Storage   │ │  Network    │ │   Power     │                │
│  │  │ Controller  │ │ Interface   │ │ Management  │                │
│  │  │   (607d)    │ │   (607e)    │ │   (607f)    │                │
│  │  └─────────────┘ └─────────────┘ └─────────────┘                │
│  └─────────────────────────────────────────────────────────────────┘
│                                ↕
│  ┌─────────────────────────────────────────────────────────────────┐
│  │                 PHYSICAL HARDWARE (608)                         │
│  │                                                                 │
│  │  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐                │
│  │  │   Central   │ │  Graphics   │ │   Tensor    │                │
│  │  │ Processing  │ │ Processing  │ │ Processing  │                │
│  │  │ Unit (CPU)  │ │ Unit (GPU)  │ │ Unit (TPU)  │                │
│  │  │   (609a)    │ │   (609b)    │ │   (609c)    │                │
│  │  └─────────────┘ └─────────────┘ └─────────────┘                │
│  │                               │                                 │
│  │  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐                │
│  │  │   System    │ │   Storage   │ │   Network   │                │
│  │  │   Memory    │ │  Devices    │ │ Hardware    │                │
│  │  │  (RAM/HBM)  │ │  (SSD/NVMe) │ │ (NIC/IB)    │                │
│  │  │   (609d)    │ │   (609e)    │ │   (609f)    │                │
│  │  └─────────────┘ └─────────────┘ └─────────────┘                │
│  └─────────────────────────────────────────────────────────────────┘
│
```

```
│
│
│  SPECIALIZED ACCELERATION (610)
│
│  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐
│
│  │    FPGA     │ │   Custom    │ │  Protocol   │ │ Quantum     │
│
│  │ Protocol    │ │   ASIC      │ │ Processing  │ │ Processing  │
│
│  │Accelerator  │ │ Chips       │ │   Units     │ │   Unit      │
│
│  │  (611a)     │ │  (611b)     │ │  (611c)     │ │  (611d)     │
│
│  └─────────────┘ └─────────────┘ └─────────────┘ └─────────────┘
│
└──────────────────────────────────────────────────────────────────────
┘

Key Hardware Optimization Features:
(612) Protocol-Native Hardware Acceleration
(613) Zero-Copy Memory Management for Large Context Windows
(614) Distributed Processing Across Heterogeneous Hardware
(615) Real-time Hardware Resource Optimization
(616) Hardware-Aware Protocol Routing and Load Balancing
```

FIGURE 7 - CROSS-CONTEXT LEARNING FLOW

```
┌──────────────────────────────────────────────────────────────────────
┐
│                   CROSS-CONTEXT LEARNING MODULE
│
│                              (701)
│
├──────────────────────────────────────────────────────────────────────
┤
│
│
│  CONTEXT SOURCES (702)
│
│  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐
│
│  │  Protocol   │ │    User     │ │  System     │ │ External    │
│
│  │ Interactions│ │ Behaviors   │ │Performance  │ │ Knowledge   │
│
│  │   (703a)    │ │   (703b)    │ │   (703c)    │ │   (703d)    │
│
│  └─────────────┘ └─────────────┘ └─────────────┘ └─────────────┘
│
│         │                │                │               │
```

```
│
│         ↓               ↓               ↓               ↓
│
│  ┌──────────────────────────────────────────────────────────────────┐
│  │              EXPERIENCE AGGREGATION (704)                        │
│  │  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐                 │
│  │  │ Pattern     │ │ Outcome     │ │ Context     │                 │
│  │  │ Extraction  │ │ Analysis    │ │ Clustering  │                 │
│  │  │   (705a)    │ │   (705b)    │ │   (705c)    │                 │
│  │  └─────────────┘ └─────────────┘ └─────────────┘                 │
│  └──────────────────────────────────────────────────────────────────┘
│                                ↓
│  ┌──────────────────────────────────────────────────────────────────┐
│  │             KNOWLEDGE SYNTHESIS (706)                            │
│  │  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐                 │
│  │  │ Federated   │ │ Semantic    │ │ Causal      │                 │
│  │  │ Learning    │ │ Embedding   │ │ Inference   │                 │
│  │  │   (707a)    │ │   (707b)    │ │   (707c)    │                 │
│  │  └─────────────┘ └─────────────┘ └─────────────┘                 │
│  └──────────────────────────────────────────────────────────────────┘
│                                ↓
│  ┌──────────────────────────────────────────────────────────────────┐
│  │            PRIVACY-PRESERVING STORAGE (708)                      │
│  │  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐                 │
│  │  │ Differential│ │ Homomorphic │ │ Federated   │                 │
│  │  │ Privacy     │ │ Encryption  │ │ Knowledge   │                 │
│  │  │   (709a)    │ │   (709b)    │ │ Base (709c) │                 │
│  │  └─────────────┘ └─────────────┘ └─────────────┘                 │
│  └──────────────────────────────────────────────────────────────────┘
```

```
│
│                                ↓
│
│  ┌─────────────────────────────────────────────────────────────────┐
│  │               ADAPTIVE OPTIMIZATION (710)                       │
│  │  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐                │
│  │  │ Protocol    │ │ Resource    │ │ Intent      │                │
│  │  │ Tuning      │ │ Allocation  │ │ Prediction  │                │
│  │  │   (711a)    │ │   (711b)    │ │   (711c)    │                │
│  │  └─────────────┘ └─────────────┘ └─────────────┘                │
│  └─────────────────────────────────────────────────────────────────┘
│
│                                ↓
│
│  LEARNING FEEDBACK LOOPS (712)
│  ┌─────────────────────────────────────────────────────────────────┐
│  │                                                                 │
│  │ Protocol A ←──→ Cross-Context ←──→ Protocol B                   │
│  │   (713a)        Knowledge Base        (713b)                    │
│  │      ↕              (714)                ↕                      │
│  │ User Context ←──→ Shared Learning ←──→ System Context           │
│  │   (713c)         Repository (715)       (713d)                  │
│  │                                                                 │
│  │     ↓                   ↓                   ↓                   │
│  │  Improved           Enhanced            Optimized               │
│  │  Protocol          Intent Cls.        Resource Mgmt             │
│  │  Efficiency         Accuracy            Performance              │
│  │   (716a)            (716b)               (716c)                 │
│  └─────────────────────────────────────────────────────────────────┘
│
│                                ↓
│
│  ┌─────────────────────────────────────────────────────────────────┐
```

```
│
│  │                SYSTEM ENHANCEMENT (717)                          │
│
│  │                                                                  │
│
│  │  • Automatic Protocol Optimization Based on Usage Patterns      │
│
│  │  • Context Assembly Improvement Through Cross-Domain Learning   │
│
│  │  • Intent Classification Refinement via Federated Training        │
│
│  │  • Resource Allocation Enhancement Through Performance Analytics│
│
│  │  • Privacy-Preserving Knowledge Sharing Across Contexts         │
│
│  │                                                                  │
│
│  └──────────────────────────────────────────────────────────────────┘
│
└──────────────────────────────────────────────────────────────────────
┘

Key Innovation Aspects:
(718) Cross-Protocol Knowledge Transfer Without Data Sharing
(719) Federated Learning Preserves User Privacy While Enabling System-Wide
Optimization
(720) Real-time Adaptation Based on Multi-Context Performance Metrics
(721) Bidirectional Learning Between Protocol Efficiency and User
Satisfaction
```

END OF PATENT APPLICATION

This provisional patent application contains the complete specification for the Protocol-as-Kernel Architecture for AI Operating Systems with Zero Static Configuration, including detailed descriptions and technical drawings suitable for USPTO filing.