**Akka framework based on the Actor model for executing distributed Fog Computing applications**

**Summary:**

**Introduction**

The Internet of Things has traditionally relied on a cloud-centric model where sensor data is collected locally and sent to distant cloud servers for processing. However, this approach suffers from significant latency issues, especially for applications requiring real-time responses. Fog computing emerged as a solution by moving some computational tasks closer to the network edge, but there's still a lack of ideal frameworks for supporting parallel and fault-tolerant execution on resource-constrained devices. The authors propose using the Akka framework, a reference implementation of the Actor Model, to address these challenges. This framework offers concurrent computation capabilities with built-in resiliency and scalability features, making it well-suited for distributed Fog applications. This research paper presents a novel approach to executing distributed Fog Computing applications using the Akka framework, which is based on the Actor Model. The authors address the limitations of traditional cloud-centric IoT systems by proposing a framework that enables parallel and fault-tolerant processing on resource-constrained edge devices. The paper demonstrates how this approach can significantly improve the performance and reliability of IoT applications while reducing latency.

**Related Work**

The authors review existing literature on Fog computing models and frameworks, noting that while there has been significant research interest in edge computing approaches, most existing solutions fall short in key areas. Current frameworks often fail to support true parallel processing and don't adequately handle fault tolerance in resource-constrained environments. The paper examines various approaches including mobile edge computing, cloudlets, and existing actor-based frameworks, highlighting how each addresses different aspects of the problem but none provide a complete solution. The authors particularly note that while several actor-based frameworks exist for IoT applications, most fail to leverage the full potential of the Actor Model for efficient resource utilization and fault tolerance in Fog networks.

**Akka Toolkit and Actor Programming Model**

The Actor Model represents a fundamental paradigm for concurrent computation where actors are isolated computational units that communicate exclusively through message passing. Each actor maintains its own state and processes messages sequentially, eliminating concerns about shared memory and race conditions. Actors can create other actors, send messages, and determine their behavior for processing future messages. This model naturally supports fault tolerance through actor hierarchies where parent actors supervise their children and can handle failures gracefully.

The Akka toolkit provides a robust implementation of the Actor Model with additional features tailored for distributed systems. It offers modules for clustering, remoting, and persistence, making it suitable for building complex distributed applications. Key features include automatic load balancing, location transparency, and built-in supervision strategies for fault handling. The framework supports both Scala and Java, providing high-level abstractions for parallelism and concurrent processing that are essential for Fog computing scenarios.

**Case Study: WSN-aware IoT Scenario**

The paper presents a comprehensive case study based on a Wireless Sensor Network (WSN) scenario to demonstrate the limitations of current approaches. In this scenario, clusters of wireless sensors collect environmental data and transmit it through gateway devices to a cloud-based Data Analytics Server (DAS). The system performs weather forecasting using ARIMA models on the collected data. This architecture represents a typical cloud-centric IoT deployment where all processing occurs in centralized servers.

The case study reveals several critical limitations of the centralized approach. First, the round-trip time for data transmission to cloud servers introduces unacceptable delays for real-time applications. Second, the system lacks resilience - if the cloud connection fails, no processing can occur. Third, the architecture doesn't utilize available computational resources at the edge of the network. These limitations motivate the need for a distributed Fog computing solution that can process data locally while maintaining the benefits of cloud computing when needed.

**Proposed Architecture**

The authors transform the centralized WSN scenario into a distributed Fog computing architecture using the Akka framework. The new architecture consists of three main clusters: the IoT Cluster, Master Cluster, and Worker Cluster. The IoT Cluster contains sensor devices and a cluster manager (gateway) that collects data using MQTT protocol. The Master Cluster, implemented as an Akka Cluster Singleton, manages the overall system state and distributes tasks to available workers. The Worker Cluster consists of multiple Fog nodes that process data in parallel using Akka's clustering and sharding mechanisms.

The architecture employs Docker containerization for dynamic deployment and management of distributed applications. Each component is packaged as a Docker container, enabling easy deployment across heterogeneous Fog devices. The system uses Akka's cluster-aware routing to distribute processing tasks efficiently among worker nodes. Fault tolerance is achieved through Akka's supervision strategies, where failures in one actor don't affect others. The architecture also supports a Shared Fog Node (SFN) concept where multiple gateway devices can share worker nodes for better resource utilization.

**Empirical Evaluation**

The evaluation setup consisted of Raspberry Pi 3 devices serving as Fog nodes, connected via WiFi. The researchers tested the system with varying numbers of sensors (2-30) generating data following a normal distribution. The evaluation focused on two main metrics: processing time and load distribution. Processing time measurements showed that while latency increases with the number of sensors, the distributed approach scales significantly better than centralized processing. The system efficiently handled up to 30 concurrent sensors on modest hardware.

Load distribution analysis revealed that the Akka framework effectively balances work across available nodes. The master cluster successfully distributed tasks based on node availability and current load. When using multiple worker clusters with shared nodes, the system demonstrated good resource utilization with the SFN handling overflow from either cluster. The results validate that the Actor Model with Akka provides practical benefits for Fog computing, including reduced processing time and better resource utilization compared to traditional approaches.

**Conclusion and Future Work**

The paper successfully demonstrates that the Akka framework based on the Actor Model provides an effective solution for executing distributed Fog computing applications. The approach addresses key challenges including parallel processing on resource-constrained devices, fault tolerance, and dynamic resource allocation. The experimental results confirm that the framework can significantly reduce processing delays while efficiently utilizing available edge resources. The use of Docker containerization adds practical deployment benefits, making the solution viable for real-world IoT scenarios.

Future work directions include extending the framework to support device mobility, as current implementation assumes static Fog nodes. The authors also plan to integrate the framework with workflow management systems and address interoperability with heterogeneous IoT devices. Additionally, they aim to implement the solution in the STEP-ONE testbed for more comprehensive evaluation under various network conditions and failure scenarios. These enhancements will further improve the framework's applicability to diverse IoT and Fog computing use cases.