Load Balancing in Adaptive Fog Computing: Research Problems and Solutions Framework

Narek Naltakyan
National Polytechnic University of Armenia
Yerevan, Armenia
e-mail: nareknaltakyan1@gmail.com

***Abstract*—Fog computing environments present unique challenges, one of which is load balancing. Problems arise here due to the geographical distribution of nodes and different computational capabilities. This paper presents a comprehensive framework designed to solve load balancing problems in adaptive fog computing systems through dynamic spatial responsibility allocation and hierarchical request distribution. It is recommended to use a multi-tiered load balancing architecture that combines geographic area adaptation, cooperative congestion management mechanisms, and threshold scaling. This framework presents three main strategies.**

1. **based on request density -> dynamic area resizing**
2. **for high demand scenarios -> horizontal scaling with load balancer deployment**
3. **through master node coordination for extreme load conditions -> hierarchical congestion management.**

***Keywords*—Load Balancing, Adaptive Fog Computing, Dynamic Area Management, Hierarchical Overflow, Edge Computing, Resource Orchestration.**

I. INTRODUCTION

Currently, the exponential growth of Internet of Things (IoT) devices poses unprecedented challenges in managing the computational load of distributed fog computing infrastructures [1]. Known solutions designed for centralized cloud environments are not able to solve the problem of fog computing. Load balancing in fog computing must consider:

1. The geographic location of services
2. Computing resources
3. Network latency constraints
4. Movement patterns of IoT devices [2]

Existing solutions typically use static load balancing strategies that cannot adapt to the dynamic nature of edge environments, leading to problems [3].

1) 1.1 Problem Statement

There are several load-balancing critical challenges that face fog computing architectures [4]:

1. **Geographic Load Imbalance:** The geographical distribution of requests can be uneven and lead to "hot spots" in the system, which can lead to nodes that are very busy and nodes that do not receive any requests at all [5].
2. **Static Resource Allocation:** Each node is assigned a fixed area of responsibility, which prevents it from changing depending on the load [6].
3. **Cascade Failure Risk:** Overloaded nodes have no mechanisms to relieve congestion [7].
4. **Limited Coordination:** Nodes operate in isolation, which prevents load sharing with neighboring nodes [8].
5. **Scaling Inefficiency:** Due to the lack of intelligent mechanisms that could distribute the load in real time and adjust infrastructure based on demand [9].

2) 1.2 Research Contributions

This paper presents a framework for adaptive fog computing that is an innovative solution for load balancing:

- **Dynamic Area-Based Load Distribution:** Automatically adjust the spatial dimensions of a node's area of responsibility based on workload.
- **Threshold-Driven Scaling Strategy:** When the area of responsibility has shrunk to a minimum size and the load does not decrease, then another node should be added to the area of responsibility.
- **Hierarchical Overflow Management:** Cooperative request processing with neighboring nodes via the master node.
- **Adaptive Decision Framework:** A comprehensive decision algorithm that selects the most optimal of a given set of options for load balancing at time X, based on system resources and request requirements.

https://doi.org/10.51408/csit2025_83

II. Hierarchical Load Balancing Architecture

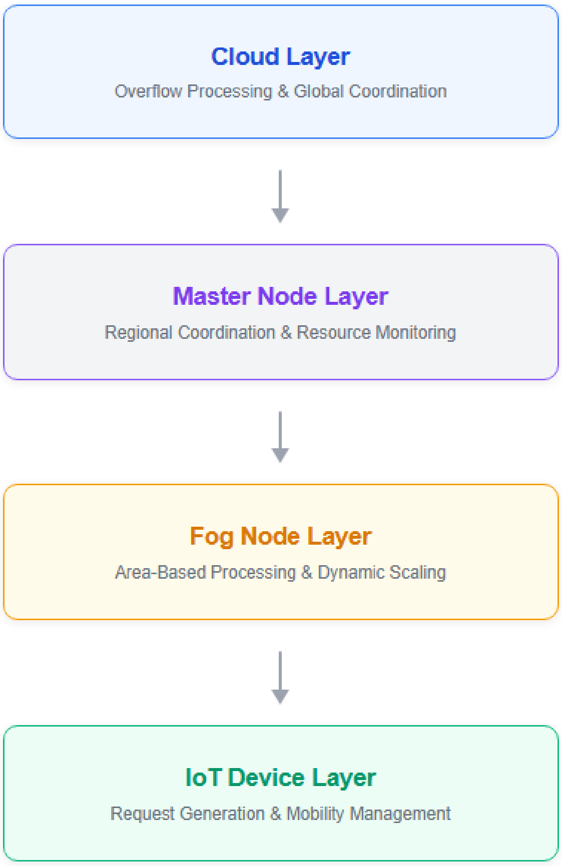

Figure 1: Hierarchical Load Balancing Architecture

1) 2 Core Components

a) 2.1 Area Manager

The area manager is responsible for dynamically adjusting the geographic area of responsibility of each node based on load metrics [10]:

- **Expansion Controller:** Increases the size of the area when the load is below a specified threshold
- **Contraction Controller:** Decreases the size of the area when the load exceeds a specified threshold or capacity
- **Boundary Coordinator:** Manages border incidents: smooth transfers of areas

b) 2.2 Load Monitor

Continuously tracks system metrics to inform load balancing decisions [11]:

- Request rate (requests/second)
- Processing latency (milliseconds)
- Queue length (pending requests)
- Resource utilization (CPU, memory, bandwidth)

c) 2.3 Scaling Orchestrator

Manages infrastructure scaling based on load conditions [12]:

- **Horizontal Scaling:** Deploys additional fog nodes and load balancers
- **Vertical Scaling:** Adjusts resource allocation within existing nodes
- **Geographic Scaling:** Redistributes areas across available nodes

d) 2.4 Overflow Manager

Handles requests that exceed local capacity [13]:

- **Local Overflow:** Redirects to load balancer when available
- **Regional Overflow:** Coordinates with master node for neighbor assistance
- **Cloud Overflow:** Escalates to cloud when fog capacity exhausted

2) 2.5 Load Balancing States and Transitions

The system operates in five distinct states, transitioning based on load conditions [14]:

Table 2: System States and Transition Conditions

State	Description	Load Range	Active Mechanisms	Transition Triggers
Idle	Minimal load, maximum area	0-20%	Area at maximum	Load > 20% → Normal
Normal	Balanced load and area	20-60%	Dynamic area adjustment	Load > 60% → High
High	Elevated load, contracting area	60-80%	Area contraction, preparation for scaling	Load > 80% → Critical
Critical	Near capacity, minimum area	80-95%	Load balancer deployment, horizontal scaling	Load > 95% → Overflow
Overflow	Exceeds local capacity	>95%	Master node coordination, cloud offloading	Load < 80% → High

3) 2.6 Request Flow Architecture

The framework implements intelligent request routing based on current system state [15]:

1. **Initial Request Reception:** IoT device sends request to assigned fog node

2. **Load Assessment:** Fog node evaluates current capacity
3. **Processing Decision:**
    - If capacity available → Process locally
    - If at capacity but area > minimum → Contract area and process
    - If at minimum area and capacity → Deploy load balancer
    - If overwhelmed → Invoke overflow procedures

III. Load Balancing Algorithms

1) 4.1 Dynamic Area Adjustment Algorithm

The core algorithm for managing geographic responsibility areas, inspired by distributed systems research [16]:

Algorithm 1: DynamicAreaAdjustment(node, currentLoad, requestRate)

Input: Fog node, current load percentage, request arrival rate

Output: Updated area boundaries

```
1: calculateOptimalArea(node)
2: IF currentLoad > THRESHOLD_HIGH AND area > AREA_MIN THEN
3:    newArea ← area × CONTRACTION_FACTOR
4:    redistributeBoundaries(newArea)
5:    notifyAffectedDevices()
6: ELSE IF currentLoad < THRESHOLD_LOW AND area < AREA_MAX THEN
7:    newArea ← area × EXPANSION_FACTOR
8:    IF canExpand(newArea) THEN
9:       redistributeBoundaries(newArea)
10:      notifyAffectedDevices()
11:   END IF
12: ELSE IF currentLoad > THRESHOLD_CRITICAL AND area ≤ AREA_MIN THEN
13:   triggerScalingProcedure()
14: END IF
15: RETURN updatedBoundaries
```

2) 4.2 Intelligent Scaling Decision Algorithm

Determines when and how to scale infrastructure [17]:

Algorithm 2: ScalingDecision(node, loadMetrics, neighborStatus)

Input: Fog node, load metrics, neighbor availability

Output: Scaling action

```
1: IF sustainedHighLoad(loadMetrics, TIME_WINDOW) THEN
2:    IF area == AREA_MIN AND loadBalancerAvailable() THEN
3:       deployLoadBalancer()
4:       addFogNode()
5:       redistributeLoad()
6:    ELSE IF neighborsAvailable(neighborStatus) THEN
7:       requestNeighborAssistance()
8:    ELSE
9:       initiateCloudOffload()
10:   END IF
11: END IF
12: RETURN scalingAction
```

3) 4.3 Hierarchical Overflow Management Algorithm

Coordinates overflow handling through master node [18]:

Algorithm 3: HierarchicalOverflow(request, sourceNode, masterNode)

Input: Overflow request, source fog node, regional master node

Output: Request handling decision

```
1: masterNode.receiveOverflowRequest(request, sourceNode)
2: availableNodes ← masterNode.findAvailableNeighbors(sourceNode)
3: IF availableNodes ≠ Ø THEN
4:    targetNode ← selectOptimalNode(availableNodes, request)
5:    IF targetNode.canAccept(request) THEN
6:       forwardRequest(request, targetNode)
7:       updateLoadStatistics()
8:    ELSE
9:       GOTO line 10
10:   END IF
11: ELSE
12:   IF cloudAvailable() THEN
13:      offloadToCloud(request)
14:   ELSE
15:      queueRequest(request)
16:   END IF
17: END IF
18: RETURN handlingDecision
```

IV. Conclusion

This paper presented a framework for load balancing in adaptive fog computing environments using dynamic zonal responsibility distribution and hierarchical congestion management. Our approach overcomes the limitations of existing solutions by integrating three mechanisms: (1) dynamic adjustment of the geographic area based on load conditions, (2) intelligent scaling of the infrastructure by deploying a load balancer, and (3) coordinated congestion management by a master node architecture.

Our work provides a foundation for the creation of adaptive fog computing systems that respond to dynamic load conditions by intelligent geographic and infrastructure adaptation. In the ever-evolving field of fog computing, especially with the support of mission-critical IoT applications, the load balancing mechanisms presented here will become more important for providing reliable, efficient, and scalable edge computing services.

Future research will focus on predictive load management on the application of machine learning for, the development of privacy-preserving federated load balancing mechanisms, and the expansion of the framework to support new edge AI workloads. The issues identified in this study provide great opportunities for improving load balancing in fog computing.

REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," *Proceedings of the MCC Workshop on Mobile Cloud Computing*, pp. 13-16, 2012.

[2] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," *Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management*, pp. 1222-1228, 2017.

[3] X. Xu, S. Fu, Q. Cai, W. Tian, W. Liu, W. Dou, and A. X. Liu, "Dynamic resource allocation for load balancing in fog environment," *Wireless Communications and Mobile Computing*, 2018.

[4] F. H. Rahman, T. W. Au, S. S. Newaz, and W. S. Suhaili, "A location-aware task offloading framework for fog computing environment," *Neural Computing and Applications*, vol. 31, no. 8, pp. 3373-3387, 2019.

[5] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289-330, 2019.

[6] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," *Internet of Everything*, pp. 103-130, 2018.

[7] S. P. Singh, A. Nayyar, R. Kumar, and A. Sharma, "Fog computing: from architecture to edge computing and big data processing," *The Journal of Supercomputing*, vol. 75, no. 4, pp. 2070-2105, 2019.

[8] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278-289, 2018.

[9] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416-464, 2018.

[10] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, "Secure and sustainable load balancing of edge data centers in fog computing," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 60-65, 2018.

[11] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, "Optimized IoT service placement in the fog," *Service Oriented Computing and Applications*, vol. 11, no. 4, pp. 427-443, 2017.

[12] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757-6779, 2017.

[13] V. B. Souza, X. Masip-Bruin, E. Marín-Tordera, W. Ramírez, and S. Sánchez, "Towards distributed service allocation in fog-to-cloud scenarios," *IEEE Global Communications Conference*, pp. 1-6, 2016.

[14] A. Brogi and S. Forti, "QoS-aware deployment of IoT applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185-1192, 2017.

[15] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171-1181, 2016.

[16] X. Q. Pham and E. N. Huh, "Towards task scheduling in a cloud-fog computing system," *18th Asia-Pacific Network Operations and Management Symposium*, pp. 1-4, 2016.

[17] K. Intharawijitr, K. Iida, and H. Koga, "Analysis of fog model considering computing and communication latency in 5G cellular networks," *IEEE International Conference on Pervasive Computing and Communication Workshops*, pp. 1-4, 2016.

[18] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Performance Evaluation*, vol. 91, pp. 205-228, 2015.