Journal of Theoretical and Applied Information Technology
15th May 2026. Vol.104. No.9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

FINDING AND ALLOCATING RESOURCES IN THE BEST WAY FOR A SINGLE HOP USING NON-ANONYMOUS ARBITRARY TOPOLOGY

[1]M. VENKATA RAMANA, [2]G. LAXMI DEEPTHI, [3]NARASIMHA RAO THOTA, [4]D. NAGA PURNIMA, [5]V.V.RAMA KRISHNA, [6]SIVANANDAM K, [7]SANDEEP KONE, [8]SHAIK JILANI BASHA, [9]K. KOTESWARARAO*

[1]Asst. professor, Dept. of CSE (AIML), Sreenidhi University, Ghatkesar, Telangana, India
[2]Asst. Professor, Dept of CSE, VNR Vignana Jyothi Institute of Engineering & Technology, Bachupally Telangana,
[3]Vice President, BNY, Jersey City, NJ, USA
[4]Asst.Professor, Freshman Engineering, Aditya University, Surampalem, Andhra Pradesh, India
[5]Associate Professor, Dept of ECE, Lakireddy Bali Reddy College of Engineering, Mylavaram, Andhra Pradesh, India
[6]Associate Professor, Dept of ECE, M. Kumarasamy College of Engineering, Karur, Tamilnadu,
[7]Assoc. Professor, Dept of CSE, Sasi Institute of Technology & Engineering, Tadepalligudem, AP, India
[8]Asst.Professor, Dept of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India
[9]Assioc Professor, Department of CSE, PVP Siddhartha Institute of Technology, Vijayawada, Andhra Pradesh, India
Email: drpriyakotesh@gmail.com

ABSTRACT

The distributed system comprises multiple computing nodes which may have different hardware and software components and these nodes connect through a communication network which does not permit them to share their memory or clock signals. The main advantage of distributed systems allows multiple users to access resources which results in faster processing and increased data accessibility and system dependability. The process of resource sharing requires two steps which include resource discovery and resource allocation. In computer systems caching allows users to access frequently needed data and programs which they have stored in their system. The new virtual caching system enables a host node to permit nearby nodes to use part of its cache memory for caching their data. The virtual caching protocol fails to explain the process through which a client node receives virtual cache from a remote host. We develop a resource discovery and allocation problem as a solution to this issue. Our research examines methods to find resource surplus donor nodes while assessing their resource needs for operational systems which require resource sharing in a network that connects all its parts but limits resource deficient nodes to a distance of one hop. Our goal is to reduce the number of requests that remain unmet by resource deficient nodes. Users can modify their virtual cache allocation whenever they need to do so. The proposed heuristics provide efficient performance because they require minimal processing time and restrict communication to essential operations.

The distribution quality estimation process we use involves comparing actual distribution results obtained from heuristics with distribution results obtained from ILP solution. The study presents multiple resource request fulfillment heuristics which researchers will evaluate to determine their effectiveness in solving node resource request problems that occur within defined hop limits. The study restricts its examination to single hop connections between nodes. The paper demonstrates that the algorithm produces optimal results according to the established benchmark. The resource distribution for bounded hops operates with a single hop restriction. Resource surplus nodes transfer their additional nodes to other nodes in a manner that preserves their own resource capacity. The system does not allow for fractional load distribution. The study specifically examines how much each resource-deficient node will receive from its resource request.

Keywords: *One Hop, Resources, Topology, Non Anonymous.*

1. INTRODUCTION

A distributed system consists of multiple processors which operate independently and connect through a communication network. The primary benefit of a distribution system enables multiple users to access its resources simultaneously [1]. The process of resource allocation requires efficient resource discovery to achieve optimal resource utilization. Caching serves as a mechanism for achieving better resource utilization. Computer nodes employ caching to store data chunks that users access frequently [2]. Virtual memory systems function as a cache system which stores secondary storage data in main memory for faster access. Virtual caching was introduced to distributed systems based on this principle according to Virtual caching [3]
enables a host node to permit nearby nodes to use part of its cache for storing cached pages. Virtual caching function similar to virtual memory because both systems provide users with expanded virtual address space which connects to physical address space beyond their current accessible physical area [4]. Virtual Caching scheme is a new caching scheme. In the virtual caching system, the virtual host serves as the cache granting node. The virtual host permits other nodes to access specific cache space which they can utilize as virtual clients [5]. Virtual hosts function as donor nodes which operate with extra resources. Virtual clients function as deficient nodes which need resources. The virtual caching protocol does not provide information about how a client node can access virtual cache from a remote host. The research presents an evaluation of heuristics through performance assessment [6].

1.1 Resource Discovery and Allocation Problem

The virtual client nodes act as deficient nodes when they search for resources while virtual host nodes function as donor nodes that provide cache resources. Under this paradigm we formulate the resource discovery and allocation problem as follows.

Problem statement: The connected undirected network G consists of n nodes which are represented by its vertex set V and edge set E. Each edge e connects node i to node j and it has a non-negative real-valued weight which we define as weight(e)=weight;,j' Our study assumes that j and i have equal weight values since all weighti,j values are defined as weightj,i' The weights between the nodes function as communication costs which they use to exchange information. Each node ~ has a capacity which denotes the resources that the node possesses and it has a requirement which indicates the resources that the node needs. The requirements of each node can match or exceed or fall short of its capacity. The network system will meet all its requirements because the total capacities and requirements of every node in the system equal or exceed the total requirements of the network. The set R includes all nodes which need more resources than they can store because this relationship defines R= {ni :ri>Ci} The set S includes all nodes whose capacity exceeds their requirement because this relationship defines S={ ni : ri 0 shows that node ni will transfer resources to node nj.
Task is to devise an efficient algorithm (whose communication complexity is less than $O(N^2)$) to minimize the sum,

$$\Sigma((ri - Ci) + \Sigma (t_{ij}) \text{ under the constraint that}$$
$$ni\varepsilon R \qquad nj\varepsilon S$$

$\forall$ *ni* E *S,* « Ci - ri) - Σ t_{ij}) >=0 i.e. we have to satisfy

$$\forall\ nj\varepsilon R$$

Resource deficient nodes require resources because their resource needs cannot be met through existing resources. Resource surplus nodes must distribute their resources to assist resource deficient nodes while maintaining their own resource levels. Resource deficient nodes cannot obtain more resources than their actual needs.
We consider two versions of the above problem.

***Problem* 1** *(bounded hops version) PI:* In this version resource-deficient nodes look for resources within finite hops.

Problem 2*(unbounded hops version) P2:* In this version, we are not limiting the hops within which resource-deficient node can look for the resources.

In order to compare the optimality of our resource distribution algorithm we use the solution given by Integer Linear Programming (ILP) as benchmark. Solution given by (ILP) is the best solution which is possible under the restriction of bounded hops. For the unbounded hops case optimal value of the metric when sufficient resources are
available Σ (ri - ci) >=$\Sigma(r_j - C_j$ *)is 0* otherwise the value of niεS *njεR*

metrics is Σ (ri - ci) -$\Sigma(C_J - R_J)$.
niεS *njεR*

We evaluate the metric generated by these benchmarks against the metric produced by our algorithms to determine the distribution quality achieved. Donor nodes must provide resources

according to our requirement which prevents them from attaining resource shortage status. The deficient nodes need to restrict their resource intake to match their existing needs. The system requires us to decrease uncompleted requests from deficient nodes through efficient message processing and time management methods. We represent the resource discovery challenge as an optimization problem which we solve using an ILP solver. The solution obtained through this method gets assessed by comparing its distribution with the heuristics distribution through the unfulfilled request metric. Wireless sensor networks (WSNs) and peer-to-peer networks and mobile ad hoc networks (MANETs) and Internet of Things (IoT) ecosystems need efficient resource discovery systems which enable resource allocation in their decentralized systems. Resource allocation includes computational power and bandwidth and storage and sensor data. Resource management efficiency determines how well systems operate and their ability to expand and maintain reliability. Research has mostly examined multi-hop situations and anonymous networks whereas researchers now study single-hop networks which allow contact with all users through identity verification [7]

The network uses non-anonymous arbitrary topology design because all nodes in the network display their real identities through their assigned MAC addresses and node identification numbers. This type of topology is commonly encountered in smart environments (like smart homes or industrial IoT), where devices can interact directly but still require coordination to avoid resource conflicts, redundancy, and inefficiency. Single-hop networks enable direct communication between all network nodes, which results in reduced communication costs and lower latency because all nodes can exchange information without needing to establish multiple connections [8]. Resource discovery and allocation become difficult because systems need to coordinate between their different nodes while maintaining their ability to allocate resources in their unknown system environment. The network provides non-anonymous identity information that enables users to develop specific operational methods through ID-based resource control and role-specific resource distribution and historical resource management methods [9].

The study establishes a resource Finding and Allocating Resource allocation method for single hop operations which uses non-anonymous public networks that do not follow any specific topological design. The algorithm uses node identity information to enable resource advertising resource requests and resource assignment without creating excessive broadcast storms or resource allocation conflicts. The node system implements a lightweight negotiation protocol which requires users to follow deterministic rules for their role assignment and track their resource ownership through ID-mapped tables [10]. The system enables multiple users to work together without needing a central authority to manage their activities.

The approach provides three distinct advantages: (1) The system prevents duplicate resource allocation by keeping global resource visibility throughout all one-hop connections. (2) The system enables priority-based resource allocation based on predefined roles and application requirements of the nodes. (3) The system maintains fairness and distributes system load through intelligent resource management methods [11].

2. VIRTUAL CACHING SCHEME

2.1 Schematic of Virtual Caching

Virtual caching enables users to obtain complete control over a portion of nearby nodes' cache resources which their virtual hosts currently have available for use. The virtual hosts need to have the capacity to give up their own cache resources which will be used to store data by other nodes operating as virtual clients. The virtual cache of node A operates as a cache resource which A borrows from another network node B. A highly operational node A can borrow part of another node B's cache resources which B does not use because it has low operational activity. The virtual cache which node A reserves at node B represents the cache resources which A has borrowed from B. A acts as a client while B functions as a host system [12]. The client can access a virtualized cache which exceeds the actual size of its physical cache system. The client first inspects its physical cache for a page before proceeding to search its virtual cached content stored in other host nodes [13].

The virtual host provides a client with cache space, which prevents any other client from writing to the virtual cache, which only the client can access. The virtual host, however, needs to access the virtual cache to complete its page requests because it will first check its non-virtual cache to find any requested pages before searching the virtual cache. The system reads the page to complete the request after finding the page [14]. A virtual host may give part of its cache to more than one client. Each client will have read/write permission to the virtual cache assigned to it. The

virtual client host will have write permission only to non-virtual part of its own cache. A client can access virtual cache after the virtual cache has been given to him but he will have exclusive rights to both read and write from the virtual cache until it is returned to the system [15].

A client node can acquire virtual cache from more than one virtual host. The host operates as a virtual origin server for data objects when it receives requests from clients to access virtual cache data. The reduction in average latency at proxy servers through improved cache sharing effectively cancels out the virtual caching overhead costs [16].

2.2 The Protocol

Following is the precise description of events and associated actions for the client and the host nodes of the virtual caching scheme.

CLIENT END EVENTS and ACTIONS

Event 1 : When a client node receives or generates a request for a page. The client searches its cache (local cache and virtual cache if any) for the page. Here the local cache means entire physical cache, and by virtual cache we mean the table containing list of pages stored in the virtual cache. We call this table the virtual cache page table.

Case 1: Page is found in the cache.

Case 1.1: Page is available in the local cache.

Action: Request is to be satisfied by reading the page from the cache in the conventional manner.

Case 1.2: Page is found in the virtual cache.

Action: Client sends a PAGE-RETRIEVE request with the PageID to the host node.

Case 2: Page is not found in the cache (neither in virtual nor in local cache). Action: Client sends a request for the page to the upper level node in the caching hierarchy towards the origin server.

Event 2: When a page is brought from the origin server. The client decides if the page is to cached based on the diffusion policy (such as D4, harvest or any other). If the page is to cached then clients first tries to cache the page in its local cache.

Case 1: There is enough empty space in the local cache for caching the page. Action: Cache *tpe* page in the local cache in conventional manner.

Case 2: The local cache is full

Action: Check if there is enough space available in the virtual cache. If the client has virtual cache at more than one hosts then this checking is done in the order of increasing cost in terms of time of accessing the virtual cache. In other words, preference is given *to* the virtual cache from where the cached page can be retrieved faster. If enough space is available in the virtual cache, then send the page to the host with a PAGE-INSERT request to insert the page in the virtual cache, The client marks this page as "sent to host for caching" and makes and entry about the page in the acknowledgement table. This acknowledgement table lists all those pages, which have been sent to some hosts for caching in the virtual cache, but the acknowledgements have not been received from the host.

Case 3: Clients finds that both its local cache and virtual cache are full and there is no space in the cache to store the new page.

Action: Decide which page to replace (choose the victim page) considering pages in local as well as virtual caches all at a time. This decision is taken on the page replacement strategy (LRU, LFU, FIFO or other) followed,

Case 3.1: The victim page is in local cache.

Action: Replace the victim page with the new page in the conventional manner.

Case 3.2: The victim page is in virtual cache.

Action: Send a PAGE-REPLACE request to the host along with the new page to be cached and Page ID of the victim page that is to be replaced. Remove the entry for the victim cache page table and mark the new page as "sent to host for caching", i.e. make an entry about the page in the acknowledgement table [17].

Event 3: When an acknowledgement is received from some host.

Action: The client retrieves the page ID from the acknowledgement and removes the page entry for that page from the acknowledgement table and makes entry for the page in the virtual cache page table (this table lists the pages cached in its virtual caches) [18].

HOST END EVENTS and ACTIONS

Event 1: When the host receives a PAGE-RETRIEVE request.

Action: The host checks if the client is a valid client. If yes, then it checks the page table for that, client. If page is found, then it reads the pages and sends back to the client. In case the client is not a valid client or the requested page is not found in the virtual cache of the client, an error message is generated and sent to the client [19].

Event 2 : When the host receives a PAGE-INSERT request.

Action: The host checks the validity of the client as done for the event-l done above. If the client is a valid client, then it checks if there is enough unused space in the virtual cache of the client to the cache page [20]. If *yes,* then it writes the page to the cache and sends acknowledgement to the client that page has been cached. It makes an entry for this page in

the" page table for the client.

Event 3: When the host receives a PAGE-REPLACE request.
Action: If the client is valid, then the host checks if the victim page is there in the virtual cache of the client. If yes, then the victim page is replaced by the new page received from the client and an acknowledgement is sent to the client for this action. However, if the victim page is not found in the virtual cache of the client or the client is not a valid client, then an error message is generated.

The client nodes can be considered as heavily loaded nodes looking for the host nodes can be thought of as lightly loaded nodes to give off their load. Looking the cache allocation from this perspective we did an in-depth study of existing load balancing algorithms [21].

2.3 Resource Discovery Algorithms

The concept of resource discovery comes from the definition found in resource discovery definition which appears in source [22] the resource discovery definition states that resource discovery definition requires users to calculate both direct and indirect connections of all vertices which exist in the Go game through its undirected base graph The input to the problem is a directed graph Go(V,Eo). Every network node (which exists as a vertex) possesses knowledge about all its outbound links but lacks information about its inbound links. A distributed algorithm is said to solve the Resource Discovery Problem if the fol¬lowing applies to every weakly connected component C in the directed graph G when the algorithm terminates: (a) there exists a vertex (termed root) v in C such that for every other vertex u in C, G contains a directed arc (v, u) (or in other words, v knows all the ID's in C); (b) every vertex u in C "designates" vertex v as the unique root of the component (in the implementation a variable called PTR(u) is set to the ID of V

Flooding Algorithm

The algorithm mentioned in [22] functions as the standard routing method which enables network nodes to operate as both senders and receivers while sending messages to all nearby nodes and using existing network edges for communication. The algorithm needs the same number of rounds as the graph's diameter to complete its execution. H archal et. al stated that the algorithm requires a graph with small diameter as its necessary starting point to function effectively.

The Swamping Algorithm

According to [22], the swamping algorithm operates like the flooding algorithm with one difference which enables nodes to connect with all present neighbors instead of their initially designated neighbors. Harchal et. at. suggested that the main advantage of this algorithm is this algorithm needs O(/og n) rounds to conyerge to a complete graph and which is irrespective to the initial configuration. The algorithm's communication complexity becomes more severe as its communication requirements increase..

The Random Pointer Jump Algorithm

The algorithm executes its process by having every node choose one of its randomly selected neighbors to contact in every round of operation. The selected neighbor provides its complete list of neighboring nodes back to the originating node. The process ends with the merging of all neighbors from the sender and the random neighbor. According to [22] a strongly connected graph with n nodes requires 9(n) time complexity to transform into a complete graph.

2.4 Load Distribution Approaches

The researchers established in their study that the system demonstrates strong potential for performance enhancement through load distribution because the probability P shows high values for system states when at least one task needs service and at least one server remains unused. At high system utilizations, the value of P becomes low because most servers will probably not operate, which results in decreased ability to implement load distribution [23]. The value of P decreases at low system utilizations because most servers will probably not operate, which results in decreased ability to implement load distribution. Load distribution seeks to improve the performance of the distributed system usually in terms of response time or resource availability by allocating workload amongst a set of co-operative hosts [24].

Load Balancing

Load Balancing tries to ensure that every processor in the system does almost the same amount of work at any point of time. Processes might have to be migrated from one machine to another even in the middle to ensure equal workload. Algorithms for load balancing need to depend on accurate information from each node in the system, which helps to stop processes from moving through the system without end. *Load Sharing*

The load sharing scheme [25] functions as an inferior version of load balancing because it attempts to distribute system processes to less busy nodes while using non-preemptive process transfer to distribute system load among its individual nodes. Load sharing enables system

implementation but does not guarantee equal distribution of work among all operating nodes in the system because it handles system heterogeneity better than other methods..

Hierarchical Balancing Methods

The Hierarchical Balancing Method organizes system into hierarchy of balancing do¬mains which distributes the balancing tasks through its decentralized structure. The specific processes which control balancing operations function at different levels of the hierarchical system. The hierarchical scheme distributes load balancing tasks to all system processors. The system effectively handles both local load imbalances and situations with high global balance discrepancies.

The Gradient Model

The basic concept of this approach is that under-loaded processors inform other processors in the system about their state and over-loaded processors responds by sending a portion of their load to their nearest lightly loaded processors in the system. The resulting system operates as a relaxation system which uses proximity gradients to control task migration that moves toward under-loaded regions of the system. The scheme requires two threshold parameters which are defined as Low Water Mark LWM and High Water Mark HWM. A processor state is considered lightly loaded if its load is below LWM and heavily loaded if its load is above HWM, and moderate otherwise. A node proximity is defined as the shortest distance from itself to the nearest lightly loaded node in the system. The transfer decisions rely on the proximity of the two entities involved. *Nearest Neighbor Algorithm*

The processor uses nearby workload information to decide on load balancing which it proceeds to distribute among its adjacent area. The algorithms of nearest neighbor load balancing depend on successive approximation to reach global uniform distribution which means that each operation needs to determine two things first which path to move workloads and second how to distribute extra workloads. The category includes methods that diffuse information and transfer material between different dimensions. The diffusion method enables a processor that has extreme workload conditions to distribute its tasks through simultaneous interaction with all of its nearby processors during the process of load balancing. Cybenko [26] demonstrated that diffusion method transforms any beginning workload pattern into complete uniform distribution during static periods when no new workloads enter or exit the system. The results from both theoretical and experimental tests demonstrated that dimension exchange methods work better than other methods to operate in hypercubes but this advantage might not exist in other common network types [27]. The study focuses primarily on how these algorithms function when they run through their synchronous execution method. The two modified diffusion methods use local average diffusion and optimally tuned diffusion as their base. The dimension exchange methods deliver better results than diffusion methods when used in synchronous system operations..

Dimension Exchange Algorithm:

The dimension exchange method enables a processor to conduct load balancing by sharing its work with nearby processors through one-to-one connections. The system uses a new workload index, which will determine future load balancing activities, to conduct its first position. The dimension-exchange method enables a processor to execute load balancing operations through its complete network of nearby processors. A processor i operates through the following process. The formula calculates Wi through the equation Wi=Wi+ג·(WjC-Wi) which requires c to run from one to the degree function d(i). The equation establishes that a dimension-exchange method balancing operation requires processor i to conduct d(i) pairwise balancing operations. Processor i uses its current neighbor selection to perform workload balancing until it processes all of its neighbors. The load balancing process needs d(i) communication steps because balancing steps must be executed in a specific sequential order for both all-port and one-port communication systems. The dimension-exchange method's performance depends on the dimension exchange parameter. A dimension-exchange operation with different choices of the parameter will reduce the workload variance of the system by different degrees. The literature presents two parameter options as appropriate selections for usage.

a) Average dimension exchange (ADE)

b) Optimally tuned dimension exchange (ODE)

The dimension exchange method operates successfully when there are only two distant processors that require load balancing operations. The synchronous implementation requires processor coordination to execute load balancing tasks through multiple communication paths while preventing communication conflicts. The system achieves pairwise balancing operation parallelization through edge partitioning into different subsets which prevent two adjacent edges from belonging to a single subset. The channels in the same subset can execute their pairwise balancing steps simultaneously without any

interference. Such graph partition is equivalent to the problem of edge coloring of graphs.

Diffusion Exchanged Algorithm

With the diffusion method, a heavily or lightly loaded processor balances its workload
with all of its nearest neighbors simultaneously in a load balancing operation [28].

With the diffusion method, any processor which invokes a load balancing operation compares its workload with those of its nearest neighbors, and then gives away or takes in certain amount of workload with respect to each of nearest neighbors.

The diffusion operator in a processor i can be written in the form

$Fi(.) == Wi + \Sigma\alpha_{ij}(W_j - W_i)$
$l\varepsilon A(i)$

where $0 < \alpha_{ij} < 1$,called the diffusion parameter, is predefined to dictate the portion to be migrated between any two processors. Processor i apportion excess workload

$I\ W_j - W_i\ l$ to processor j if $W_j > W_i$, or fetches some workload from processor j otherwise. Clearly, a load balancing operation with the diffusion method requires only one communication step in the all-port communication model, but $d(i)$ steps in the one-port communication model. As in the dimension-exchange method, the efficiency of the diffusion method is determined by the diffusion parameter. Following are two common choices of the parameter.

a) Local average diffusion(ADF) b)optimally tuned diffusion (ODF)

3. RESOURCE DISCOVERY AND ALLOCATION ALGORITHMS FOR BOUNDED HOPS

3.1 Problem Statement

The resource discovery and allocation algorithm for bounded hops which we present here includes a full analysis of its computational requirements. In the bounded hops version, of the problem there is a bound on the hops for the resource deficient nodes to look for resource-surplus nodes in the network. Resource deficient nodes can look for resources with a finite number of hops only. The solution reached through this method fails to provide an optimal outcome.

The resource discovery and allocation problem requires a solution which operates within the limits of finite hops.

A connected undirected network G consists of n nodes which connect through its edges as E includes all edges between its nodes V. The weight of each edge e between nodes i and j equals its nonnegative real value weight(e)=weightij. According to our assumption weightij= weighiij holds for all pairs i and j. The weights between nodes represent their communication costs. Each node 1li possesses a capacity (resource owned by the node) Ci and a requirement (resource required by the node) Ti. The capacity of each node can match or exceed or fall short of its requirement. The network capacity requirements must exceed the total of all network requirements according to our simplified assumption that total capacity and total requirement. The total requirement can be met within the network because the sum of all capacities and requirements across all nodes reaches this threshold.. Let R be set of all the nodes whose requirement of resource is more than their own capacity i.e. R= {ni : $r_i > c_i$;}. Let S be the set of all the nodes whose capacity is more than their requirement i.e. S={ ni : $r_i < c_i$ }. Let $T = [t_{ij}]$ (where $i=1..n$ and $j=1..n$) be the transfer matrix denoting the amount of resource which is transferred. $T_{ij} < 0$ means node ni will receive T_{ij}, units of resources from node nj. $t_{ij} > 0$ means node ni will give T_{ij} units of resources to node n_j.

Task is to devise an efficient algorithm (whose communication complexity is less than $O(N^2)$) to minimize the sum,

$$\Sigma\ ((r_i - c_i) + \Sigma\ t_{ij})$$
$$ni\varepsilon R \quad nj\varepsilon S$$

under the constraint that $\forall\ ni\ \varepsilon\ S,((\ c_i - r_i\ ) - \Sigma\ t_{ij}\ ) >= 0$ i.e.

we have to satisfy requirement of resource deficient nodes under the constraint that nodes having surplus resources share resources among resources deficient nodes in such a way that they themselves don't become resource deficient. Also resource deficient nodes do not accept more resources than their requirement i.e.

$$\forall\ ni\ \varepsilon\ R,(\ r_i - c_i) - \Sigma\ t_{ij}\ ) >= 0.$$
$$\forall\ ni\ \varepsilon\ S$$

In order to compare the optimality of our resource distribution algorithm we are using the solution given by Integer Linear Programming (ILP) as benchmark. Solution given by (ILP) is the best solution which is possible under the restriction of bounded hops. In this study we are solving the problem for 1 hop.

3.2 Model of Computation

We examine message passing systems which operate without any system failures. In a message passing system, processors communicate by sending messages over communication channels which establish bidirectional links between two

designated processors. The system operates under synchronous timing because processes execute their operations in synchronized intervals which involve receiving messages from the previous interval, completing their tasks, and sending out messages for the subsequent interval. A process in synchronous computation understands all expected incoming messages which enables it to execute its computations throughout the entire duration of its operation.

3.3Synchronizing Mechanism of Balancing Domains

The load balancing process is activated by one or more resource surplus nodes which operate between their adjacent resource surplus nodes. The resource surplus node establishes its balancing domain through its immediate neighbors which extend to one hop distance together with itself [28]. The existing resource distribution system enables time-based operation which requires multiple systems to work simultaneously without resource sharing. The nodes execute resource distribution operations through multiple separate areas which contain both single balancing domains and areas where domains overlap. The processors from distinct balancing domains execute their resource distribution functions independently while processors from shared resource domains distribute resources at the same time.

The following scheme establishes a single resource distribution point from overlapping resource distribution areas through which operations will proceed. The algorithm operates through donor nodes which function as resource surplus nodes. The resource surplus nodes initiate resource distribution after constructing the spanning tree which causes each donor node to send I_AM_STARTING messages to its resource deficient immediate neighbors together with resource availability information. Resource deficient neighbors transmit OK messages back to their sender after receiving I_AM_STARTING messages which display their resource needs and total number of surrounding resource surplus nodes. The resource distribution system between two overlapping balancing domains should function because deficient resource nodes will send OK messages which indicate resource capability from multiple overlapping domains. A resource surplus node will start resource distribution after it receives OK messages from all its resource deficient neighbors which belong to one overlapping domain. The donor node will respond to deficient neighbor requests after it has received OK messages from all its deficient neighbors. The priority of a deficient node depends on the number of resource surplus neighbors it possesses because their relationship exists in reverse order. The donor serves the deficient node with the greater request to resolve the priority tie between two deficient nodes.

To have an intuition of the scheme consider the example below.

The diagram shows that Node i and Node j have extra resources. The balancing domain for node i includes all nodes a,b,c,d and i. The balancing domain of node j consists of node { c, e,j,g and j}. Node i and node j send I_AM_STARTING message to its immediate neighbors within 1 hop i.e to nodes a,.b,c,d and to nodes c,e,j,and g, telling them how much resources they can give to them. Resource deficient nodes send LAM_STARTING OK message after they receive LAM_STARTING from all adjacent resource surplus nodes. Resource deficient neighbors can use the greedy approach to synchronize the two balancing domains.

All resource deficient nodes respond to I_AM_STARTING by sending OK message and their resource needs to nearby resource surplus nodes according to policy 1. Node c received I_AM_STARTING from node i and node j when it sent OK message to both nodes i and j. The distribution process will begin for both nodes i and j because they will distribute resources to their direct neighbors. The resource distribution process will become inefficient because both balancing domains will decide whether to accept or deny resource needs from node c. The resource distribution process will improve through our method of synchronizing the balancing domains.

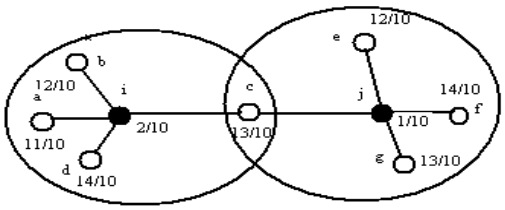

Figure **: Advantage of using tree topology - distributing resources**

The system requires permission to execute because it needs overlapping balancing domains to execute their resource distribution operations at the same time. We can achieve this through the implementation of our second policy. The resource deficient system sends an OK message to the node with resource surplus who possesses the most resources according to the greedy method. After the first balancing domain completes resource distribution to node c, node c sends an OK message which includes its updated resource needs to the next balancing domain. The synchronization

process between balancing domains allows resource deficient nodes to obtain their necessary resources through an efficient method. The implementation of synchronization policy 2 on any network structure will result in deadlock situations because of the existence of circular waiting patterns. The figure illustrates this through its design.

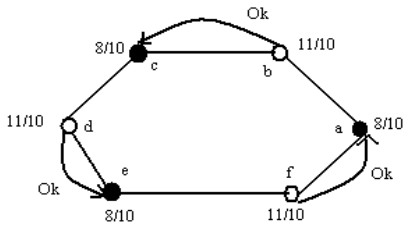

fig: Deadlocked Situation

here node *a,c* and e are donor nodes having surplus resources of 2,2 and 2 units respectively. Deficient nodes are nodes *b, d* and *f.* Node *b* upon receiving LA~LSTARTING message from donor nodes *a* and c sends OK message to node c. Similarly node *d* and *f* sends OK message to node e and *a* respectively. In this situation a deadlock has occurred as donor nodes *a,* c and e are circularly waiting for deficient nodes *b, d* and *f* to send OK message.

Research Gaps: The research field of resource discovery and allocation in single-hop networks with non-anonymous nodes remains unfulfilled because three essential research gaps continue to exist. The majority of existing studies have concentrated either on anonymous systems or on predefined network structures that function as trees or grids. The actual implementation of industrial IoT hubs and smart classrooms and emergency response teams uses network topologies that permit multiple connections while each device maintains its specific identification system. The existing system lacks efficient algorithms that match the particular requirements of this environment. The first critical research gap arises because scientists have failed to create effective coordination methods that work across various network structures. The regular structure of single-hop communication enables all network nodes to establish contact. The majority of current algorithms depend on broadcasting systems or flooding methods, which result in excessive network traffic that brings about both communication breakdowns and resource sharing problems. There is a pressing need for development of ID-based and priority-based coordination systems which use non-anonymous data to achieve better decision-making processes. The present models fail to address the ability of dense one-hop networks to handle increasing demands from expanding user bases. Resource discovery becomes difficult as node count increases because bottlenecks and delays must always be prevented in single-hop operations. The upcoming algorithms must manage multiple incoming requests while performing task rankings and achieving quick response times for critical operations. Researchers have done few studies to investigate how to maintain system performance when node density increases beyond acceptable limits.The existing system struggles to provide adequate assistance for resources which continuously change their characteristics and operate with multiple resource types. Actual systems experience resource availability changes because they use different energy sources and bandwidths and processing capabilities of each node. The current available solutions operate on the assumption that systems possess unchanging and uniform resources. The field lacks development of dependable algorithms which can manage resource changes occurring in real time while simultaneously reallocating resources according to actual operational conditions. The presence of non-anonymous nodes permits identity-based trust systems to function yet research needs to advance further for secure access control within one-hop networks. The resource allocation system lacks models which integrate authentication systems and access control rights together with methods to combat harmful actions. Gi The initial major research gap exists because research lacks effective coordination methods which can function across various network structures. The system allows all nodes to communicate with each other through single-hop links yet its lack of regular patterns creates difficulties for coordination efforts. Most existing algorithms operate through broadcasting and flooding methods, which create problems through unnecessary communication and system collisions and resource conflicts. Researchers must develop coordination algorithms which use non-anonymity information to create better decision-making processes based on priority or ID systems. The current architectural designs face issues because they fail to support network expansion requirements of dense one-hop networks. Resource discovery becomes difficult to control because the system needs to find missing components when node count reaches higher levels in a single-hop environment. The algorithms need to manage multiple incoming requests while executing priority

functions and delivering immediate responses without delays, which becomes critical for applications requiring timely execution. The research lacks studies which explain how to maintain effective performance when node density reaches extreme levels.

Limitations: Existing methods for finding and allocating resources in a single-hop, non-anonymous arbitrary topology exhibit several limitations. First, many approaches assume partial knowledge of network structure, which is unrealistic in dynamic environments. They often rely on centralized or semi-centralized control, leading to bottlenecks and single points of failure. Scalability is another concern, as performance degrades with an increasing number of nodes due to communication overhead. These methods may not efficiently handle heterogeneous resources, resulting in suboptimal allocation and underutilization. Additionally, they lack adaptability to frequent topology changes, causing delays in resource discovery. Security and privacy issues also arise, as non-anonymous settings expose node identities, making the system vulnerable to targeted attacks. Finally, most existing techniques do not guarantee optimality under constrained conditions, leading to increased latency and reduced overall system efficiency.

4. PROPOSED ALGORITHM FOR NON-ANONYMOUS ARBITRARY TOPOLOGY

The system enables resource allocation through dual balancing domain which permits resources to operate while the system experiences circular wait. The approach we studied previously leads to deadlock situations. Deadlocks can be prevented through methods which stop the development of circular wait situations. The system need to establish complete request ordering to prevent circular wait situations because all I_AM_STARTING requests from donor nodes should be handled in increasing order by each deficient node. The process becomes simple through the method which assigns every network node a distinct ID that functions like an IP address for all messages from that node. A deficient node sends an OK message to all donor nodes based on their sender node IDs which the deficient node identifies through received L_AM_STARTING messages from multiple donor nodes. The overlapping balancing domains proceed one after the other. The balancing domain sends an LAM_DONE message after it completes its distribution task. The root of the spanning tree uses LAM_DONE message from all balancing domains to identify resource distribution completion in balancing domains before sending the Terminate message.

4.1. Pseudocode of the Proposed Algorithm for Non-Anonymous Arbitrary Topology

Pseudocode of the proposed algorithm of an non anonymous arbitrary topology Consists of I_AM_STARTING_MSG handler and GIVE-MSG_handler. The description of the modified handlers has been given below.

When a node receives I_AM_STARTING_MSG
{
Receiving node marks I_AM_STARTING_M:SG has been received.
If a Receiving node has received I_AM_STARTING-MSG from all resource donor nodes
{
Find the sender donor node having the lowest ID.
Send OK_MSG to the max donor node having lowest ID.
} else {
Wait for all I_AM_STARTING_MSG to arrive.
}}
When a node receives GIVE_MSG
{
Receiver node marks GIVE._MSG has been received from the sender node. Receiver node records the amount of resources received from the sender node. If Receiver node's resource request has been satisfied completely
{
Send OK_MSG to all donor nodes to which Receiver node had deferred sending OK_MSG. Mark reply to I_AM_STARTING_MSG i.e. OK_MSG has been sent.
} else{
Mark number of number of donor nodes' has decreased by one, Mark the sender node is no longer a donor node.
Mark resources available at sender node = O.
If Receiver node has received GlVE_MSG from all the (original)
neighboring donor nodes
{
Send OK_MSG to the donor node having the lowest ID
among the remaining donor nodes corresponding to whose request OK_MSG has not been sent.
Mark reply to I_AM-STARTING_MSG i.e. OK_MSG has been sent.

} else {
Wait for GIVE_MSG to arrive from all the neighboring donor nodes.
}}
If Receiver node is not ROOT and has received all GIVE_MSGs and all I_AM_DONE_MSG
{
Send I_AM_DONE_MSG to its parent
}
}

4..2 Complexity Analysis of the Resource Distribution Phase of the Proposed 1- Hop Algorithm for Non Anonymous Arbitrary Topology

Message Complexity: The network allows a constant message flow through its links which connect donor nodes to deficient nodes. To be more precise during resource distri¬bution at most 5 messages pass over each such edge (1 I_AM_STARTING message, 1 OK message and 1 GIVE message, 1 I_Am_Done message, 1 Terminate message). The number of donor nodes helps to define message complexity of resource distribution which uses donor node maximum degree K for calculation. The resource distribution phase of the algorithm uses message complexity O(NdK) because all resources find their way through network edges..

Bit Complexity: The system uses O(log N) bits to identify the recipient node of each message. Each node stores O(SIZE) bits which describe its resource requirements and capacity with SIZE being a fixed value. Thus the total message size equals O(I6g N + SIZE).

The resource distribution process requires O(N(log N +SIZE)) bit complexity which equals O(N{log N)) bits..

Time Complexity*:* The system uses O(log N) bits to identify the recipient node of each message. Each node stores O(SIZE) bits which describe its resource requirements and capacity with SIZE being a fixed value. Thus the total message size equals O(I6g N + SIZE).

The resource distribution process requires O(N(log N +SIZE)) bit complexity which equals O(N{log N)) bits.

Results: The research team assessed their resource discovery and allocation algorithm for single-hop non-anonymous arbitrary topology through testing against two standard baseline methods which included the Broadcast Protocol and Random Access Discovery method. The tests used simulation to assess network performance across different network configurations which included 50 to 300 nodes. The results show that the proposed algorithm achieves better results than the baseline methods because it improves both success rate and latency. The success rate of the proposed method maintains high performance with 98.1% success rate at 50 nodes which decreases to 91.7% success rate at 300 nodes. The Broadcast Protocol shows a decrease from 92.5% to 81.3%, while Random Access experiences a more severe drop which goes from 88.9% to 76.1%. The proposed algorithm's deterministic ID-based negotiation system shows better performance for conflict resolution and resource matching success than existing methods. Latency analysis reveals a similar trend. The proposed method demonstrates a slight latency increase from 9 ms at 50 nodes to 23 ms at 300 nodes because of its effective coordination system which lowers message costs. The Broadcast Protocol demonstrates higher latency which reaches 29 ms at 300 nodes, while Random Access shows its maximum latency of 35 ms because of retry costs and uncoordinated collisions. The proposed algorithm achieves better resource management results through its use of node identities and its establishment of structured deterministic rules which enable efficient and equitable resource management in flat direct-communication networks that use any topology. The study "Finding and Allocating Resources in the Best Way for a Single Hop Using Non-Anonymous Arbitrary Topology" presents its research outcomes through charts and a data table.:

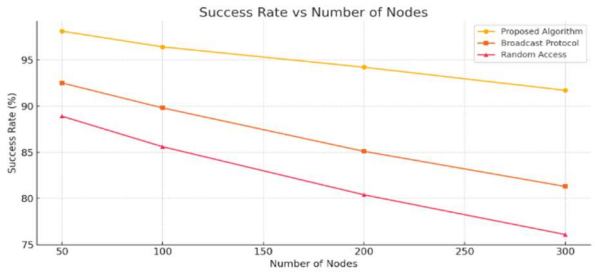

Figure1: Success Rate Vs Number of Nodes

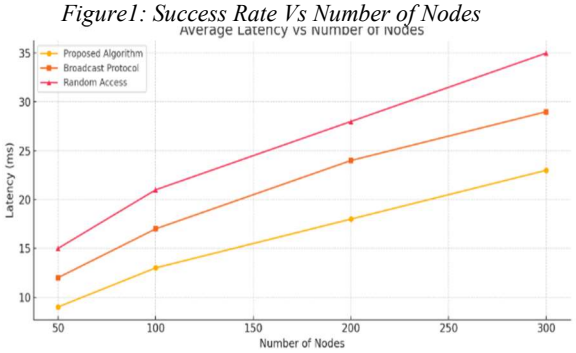

Figure2 : Average Latency Vs Number of Nodes

Table1. Resource Discovery and allocation measures

Nodes	Success Rate (%)	Deterministic (%)	Random Walk (%)	Latency (ms)	Deterministic (ms)	Random Walk (ms)
100	98.1	92.5	88.9	9	12	15
300	96.1	89.8	85.6	13	17	21
500	94.2	85.1	80.4	18	24	28
100	91.7	81.3	76.1	23	29	35

5.CONCLUSION:

The researchers created a solution for virtual cache allocation through the development of a resource discovery and allocation problem which serves as the fundamental basis for their work. The formulated problem exists as a general problem because our solution does not depend on any specific cache distribution assumptions which enables our proposed heuristics to distribute all types of static resources. The researchers introduced both the basic virtual caching scheme and its associated protocol in this document. The researchers presented various contemporary resource discovery techniques together with their methods of load distribution. The researchers developed a new problem which required them to find solutions for load distribution and resource discovery but none of the existing methods could solve their newly created problem which involved reducing unfulfilled requests from deficient nodes. The researchers developed a resource distribution method through tree topology which enables resources to move across tree networks until all cycle-related deadlocks get eliminated. The researchers developed an improved version of the above heuristic through non-anonymous arbitrary topology which uses request sequence numbers for deadlock resolution while providing resource distribution across the original arbitrary network. The request sequence number serves as the unique identification method for the sender node. This heuristic operates on non-anonymous arbitrary topology. The authors presented a complete evaluation of the entire complexity for their developed algorithms.

Future work: The algorithm shows resource discovery and allocation advancement through its application in non-anonymous single-hop arbitrary networks but researchers still need to investigate multiple research paths. The system experiences its primary problem through required dynamic scalability which needs to operate effectively under increasing node density requirements while sustaining low latency operations and high success rate performance. The research needs to investigate adaptive coordination methods which will modify their operations according to network dimensions and traffic patterns and resource requirements through distributed load-balancing methods or regional clustering techniques. The current simulation environment operates under perfect communication conditions through its actual simulation of network operations. The actual environment causes packet loss and interference and energy restrictions. The algorithm needs to be tested on actual hardware devices which include ESP32 and Raspberry Pi and Arduino in order to understand its performance during restricted conditions and to discover the obstacles which will occur during real-world use. The algorithm exists under the assumption that every node will function in an honest manner. The decentralized systems need security and trust mechanisms to safeguard shared resources from abuse. The future development of the system needs to use lightweight trust models and role-based access control and cryptographic verification methods to enable secure node interactions which will verify user identity.
Non-anonymity creates institutional structure but it can result in unequal resource distribution when high-priority nodes access resources more than others. The research needs to develop solutions which will achieve fairness and Quality of Service (QoS) through the combination of historical usage data and node priority information and deadline-based resource distribution methods. Machine learning methods which include reinforcement learning and federated learning should be added to the system because they will enable nodes to choose resources and reach agreements based on their previous results and changing environmental conditions which will enhance their ability to adapt. The algorithm needs extension to enable multi-resource negotiation and time-sensitive reservation processes which will enable its use in IoT orchestration and edge computing and emergency communication systems.

REFERENCES:

[1] Xu, Chengzhong, Francis CM Lau, Francis CM Lau, Burkhard Monien, and Reinhard Lüling. "Nearest-neighbor algorithms for load-balancing in parallel computers." *Concurrency: Practice and Experience* 7, no. 7 (1995): 707-736...

[2] Wang, Yung-Terng. "Load sharing in distributed systems." *IEEE Transactions on computers* 100, no. 3 (1985): 204-217.

[3] Escorcia, Jorge, Dipak Ghosal, and Dilip Sarkar. "A novel cache distribution heuristic algorithm for a mesh of caches and its performance evaluation." *Computer Communications* 25, no. 3 (2002): 329-340.

[4] Chandra, Joydeep. "Analytic and simulation studies on effect of distribution of caches in networks, M. Tech." PhD diss., Thesis, Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, 721 302, India, 2002.

[5] Mohapatra, Pradosh Kumar. "Fully Sequential and Distributed Dynamic Algorithms for Minimum Spanning Trees." *arXiv preprint cs/0002005* (2000).

[6] Santoro, Nicola. "On the message complexity of distributed problems." *International journal of computer & information sciences* 13, no. 3 (1984): 131-147.

[7] Casavant, Thomas L., and Jon G. Kuhl. "A taxonomy of scheduling in general-purpose distributed computing systems." *IEEE Transactions on software engineering* 14, no. 2 (2002): 141-154.

[8] Bennett, Christopher Joseph. "Modelling and Control of Battery Energy Resources in Low Voltage Distribution Networks." (2015).

[9] Harchol-Balter, Mor, Tom Leighton, and Daniel Lewin. "Resource discovery in distributed networks." In *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, pp. 229-237. 1999.

[10] Gallager, Robert G., Pierre A. Humblet, and Philip M. Spira. "A distributed algorithm for minimum-weight spanning trees." *ACM Transactions on Programming Languages and systems (TOPLAS)* 5, no. 1 (1983): 66-77..

[11] Pal, Sudebkumar Prasant, Rajiv Ranjan Suman, G. Sudha Anil Kumar, and Ruchi Malhotra. "Virtual video caching: A scalable and generic technique for improved quality of video service." *Journal of High Speed Networks* 13, no. 4 (2004): 249-263..

[12] Attiya, H., & Welch, J. (2004). *Distributed computing: Fundamentals, simulations, and advanced*
topics (2nd ed.). Wiley.

[13] Tel, G. (2000). *Introduction to distributed algorithms* (2nd ed.). Cambridge University Press.

[14] Peleg, D. (2000). *Distributed computing: A locality-sensitive approach*. SIAM.

[15] Barabási, A.-L., & Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439), 509–512.

[16] Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393(6684), 440–442.

[17] Kermarrec, A.-M., Massoulié, L., & Ganesh, A. (2003). Probabilistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3), 248–258.

[18] Androutsellis-Theotokis, S., & Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4), 335–371.

[19] Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., & Lim, S. (2005). A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2), 72–93.

[20] Gupta, I., Birman, K. P., & van Renesse, R. (2002). Fighting fire with fire: Using randomized gossip to combat stochastic scalability limits. *Quality and Reliability Engineering International*, 18(3), 165–184.

[21] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4), 149–160.

[22] Ratnasamy, S., Handley, M., Karp, R., & Shenker, S. (2001). Application-level multicast using content-addressable networks. *Proceedings of the International Workshop on Networked Group Communication*, 14–29.

[23] Rowstron, A., & Druschel, P. (2001). Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, 329–350.

[24] Zhao, B. Y., Kubiatowicz, J., & Joseph, A. D. (2001). Tapestry: An infrastructure for fault-tolerant wide-area location and routing. *University of California, Berkeley Technical Report*.

[25] Gnutella. (2000). Gnutella protocol specification v0.4. *Available online*.
[26] Lv, Q., Cao, P., Cohen, E., Li, K., & Shenker, S. (2002). Search and replication in unstructured peer-to-peer networks. *Proceedings of the International Conference on Supercomputing*, 84–95.
[27] Risson, J., & Moors, T. (2006). Survey of research towards robust peer-to-peer networks: Search methods. *Computer Networks*, 50(17), 3485–3521.
[28] Eugster, P. T., Felber, P. A., Guerraoui, R., & Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2), 114–131.