JYOTI NIVAS COLLEGE AUTONOMOUS

DEPARTMENT OF MCA

TECH-ON-TOP

E-JOURNAL ON DISTRIBUTED SYSTEM

ISSUE 7

MARCH 2021

INDEX

DISTRIBUTED COMPUTING

Distributing computing is a method of computer processing in which different parts of a computer program are run on a two or more computers that are computing with each other over a network. Each processor has its own private memory (distributed memory) and coordinate their actions by passing message to one another from any system. The components interact with one another in order to achieve a common goal. Distributing computing usually have 3 primary characteristics all components run concurrently, there is no global clock and components fail independently of each other.

TYPES OF DISTRIBUTED COMPUTING:

- **Cluster computing:** In cluster computing each computer is referred as node that are interconnected and work together to perform computationally intensive tasks.
- **Grid computing:** Multiple in dependent computing clusters which act like a grid because they are composed of resource nodes not located within a single administrative domain.
  The creation of a "virtual supercomputer" by using spare computing resources with in an organization.
- **Cloud computing:** Cloud computing is a computing paradigm shift where computing is moved away from personal computers or an individual application server to a cloud of computers. Users of the cloud only need to be concerned with the computing service being asked for, as the underlying details of how it is achieved are hidden. This method of distributed computing is done through pooling all computer resources together and being managed by software rather than a human.

ARCHITECTURE:

- **Client-server:** Architectures where smart clients contact the server for data then format and display it to the users. Input at the client is committed back to the server when it represents a permanent change.
- **Three-tier:** Architectures that move the client intelligence to a middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are three-tier.
- **N-tier:** Architectures that refer typically to web applications which further forward their requests to other enterprise services. This type of application is the most responsible for the success of application servers.
- **Peer-to-peer:** Architectures where there no special machines that provide a service or manage the network resources. Instead all responsibilities are uniformly

divided among all machines, known as peers. Peers can serve both as client and as servers.

Distributed computing is the most efficient way to achieve the optimization. In case of distributed computing the actual task is modularized and is distributed among various computer system. It not only increase the efficiency of the task but also reduce the total time required to complete the task.

REFRENCE:

- https://en.m.wikipedia.org/wiki/Distributed_computing
- https://www.ionos.com/digitalguide/server/know-how/what-is-distributed-computing/
- https://www.tutorialspoint.com/software_architecture_design/distributed_architecture.htm

ABHILASHA
(19MCA01)

FAULT TOLERANCE IN DISTRIBUTED SYSTEM

Fault tolerance refers to the ability of a system (computer, network, cloud cluster, etc.) to continue operating without interruption when one or more of its components fail.

The objective of creating a fault-tolerant system is to prevent disruptions arising from a single point of failure, ensuring the high availability and business continuity of mission-critical applications or systems.

Fault-tolerant systems use backup components that automatically take the place of failed components, ensuring no loss of service. These includes

Hardware systems that are backed up by identical or equivalent systems. For example, a server can be made fault tolerant by using an identical server running in parallel, with all operations mirrored to the backup server.

Software systems that are backed up by other software instances. For example, a database with customer information can be continuously replicated to another machine. If the primary database goes down, operations can be automatically redirected to the second database.

Power sources that are made fault tolerant using alternative sources. For example, many organizations have power generators that can take over in case main line electricity fails.

In similar fashion, any system or component which is a single point of failure can be made fault tolerant using redundancy.

Fault tolerance vs. high availability

High availability refers to a system's ability to avoid loss of service by minimizing downtime. It's expressed in terms of a system's uptime, as a percentage of total running time. Five nines, or 99.999% uptime, is considered the "holy grail" of availability.

In most cases, a business continuity strategy will include both high availability and fault tolerance to ensure your organization maintains essential functions during minor failures, and in the event of a disaster.

While both fault tolerance and high availability refer to a system's functionality over time, there are differences that highlight their individual importance in your business continuity planning.

Some important considerations when creating fault tolerant and high availability systems in an organizational setting include:

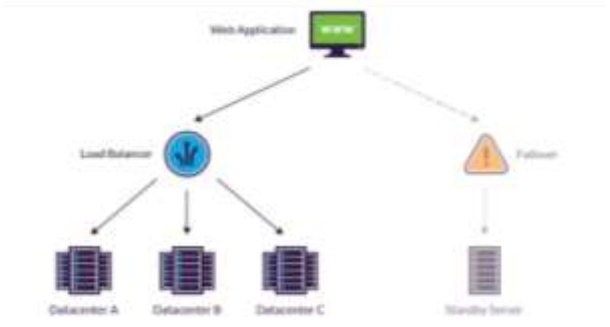
- **Downtime** – A highly available system has a minimal allowed level of service interruption. For example, a system with "five nines" availability is down for approximately 5 minutes per year. A fault-tolerant system is expected to work continuously with no acceptable service interruption.

- **Scope** – High availability builds on a shared set of resources that are used jointly to manage failures and minimize downtime.
- **Cost** – A fault tolerant system can be costly, as it requires the continuous operation and maintenance of additional, redundant components. High availability typically comes as part of an overall package through a service provider (e.g., load balancer provider).

Some of your systems may require a fault-tolerant design, while high availability might suffice for others. You should weigh each system's tolerance to service interruptions, the cost of such interruptions, existing SLA agreements with service providers and customers, as well as the cost and complexity of implementing full fault tolerance.

Load balancing and failover: fault tolerance for web applications

In the context of web application delivery, fault tolerance relates to the use of load balancing and failover solutions to ensure availability via redundancy and rapid disaster recovery.



Load balancing and failover are both integral aspects of fault tolerance.

Load balancing solutions allow an application to run on multiple network nodes, removing the concern about a single point of failure. Most load balancers also optimize workload distribution across multiple computing resources, making them individually more resilient to activity spikes that would otherwise cause slowdowns and other disruptions.

References:

1. https://avinetworks.com/glossary/faulttolerance/
2. https://www.slideshare.net/sumitjain201/fault-tolerance-in-distributed-systems

Amina S.N
19MCA02

AN OVERVIEW OF SOFTWARE DEFINED NETWORK

INTRODUCTION: Software Defined Network (SDN) is a network architectural model that allows optimization, control and programmatic management of network resources which uses application programming interface (API) and software based applications for communication on network and underlying hardware architecture. In traditional network, whenever a new device upgraded or integrated new functionality should be added along with it and also it has interoperability issues. Hence the traditional network is not flexible and programmable, to overcome SDN is introduced.

SDN: SDN plays important role in networking industry. It has network controller which forwards packets and open switches called open-flow which is a standard protocol works independent of control forward behavior and vendor equipment. Unlike traditional network, SDN has to install application when a new device is added. SDN provides a very flexible interface for creation of new services. SDN is built upon 3 main concepts they are Programmable Networks, Centralized Network Control and Data Plane Separation.

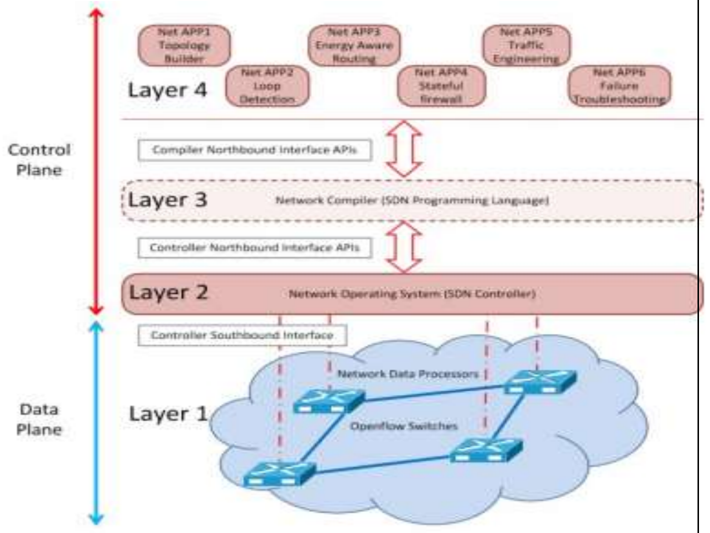

Net APP1 Topology Builder
Net APP3 Energy Aware Routing
Net APP5 Traffic Engineering
Layer 4
Net APP2 Loop Detection
Net APP4 Stateful firewall
Net APP6 Failure Troubleshooting
Control Plane
Compiler Northbound Interface APIs
Layer 3
Network Compiler (SDN Programming Language)
Controller Northbound Interface APIs
Layer 2
Network Operating System (SDN Controller)
Controller Southbound Interface
Network Data Processors
Data Plane
Layer 1
Openflow Switches

Figure 2 Layered architecture of an SDN network

CONCLUSION: SDN provides simplified and automated network management that increases the demand of network complexity and many application domains, improves policy control, scalability & remove vendor dependencies. It is only networking paradigm, an appropriate application must be used to use its benefits. SDN aids for the concepts such as IOT (internet of things), Cloud Services & Cloud Integration, Mobility and wireless devices.

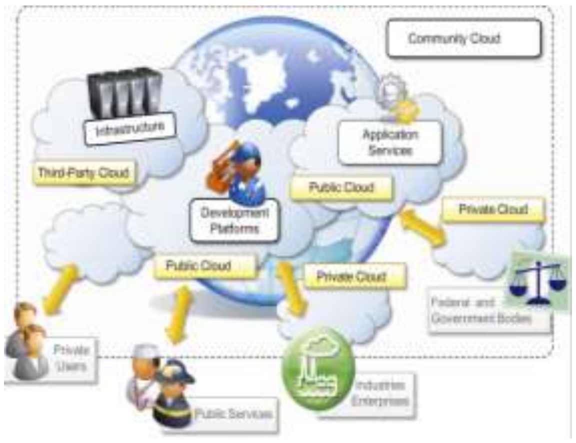
REFERENCE:

- https://www.researchgate.net/publication/312569297_Software_Defined_Networking_concepts_and_challenges
- https://www.researchgate.net/publication/267339360_Software-Defined_Networking_Challenges_and_research_opportunities_for_Future_Internet

CHAITHRA
19MCA04

CLOUD COMPUTING

Cloud Computing is one of the most significant transformation in information technology with many advantages to both companies and end users, and the main feature of Cloud Computing is combination of vast software and common hardware to provide a powerful Computing Paradigm. Cloud Computing provides an opportunity to increase efficiency and reduce costs in the IT portion of the business, by offering the possibility of using systems with state of art computing capabilities, high availability and scalability.



SERVICES OF CLOUD COMPUTING

SaaS (software as a service)

This type of public cloud computing delivers applications over the internet through the browser. The most popular SaaS applications for business can be found in Google's G Suite and Microsoft's Office 365 among enterprise applications, Salesforce leads the pack. But virtually all enterprise applications, including ERP suites from Oracle and SAP have adopted the SaaS model. Typically, SaaS applications offer extensive configuration options as well as development environments that enable customers to code their own modifications and additions.

IaaS (infrastructure as a service)

At a basic level, IaaS public cloud providers offer storage and compute services on a pay-per-use basis. But the full array of services offered by all major public cloud providers is staggering: highly scalable databases, virtual private networks, big data analytics, developer tools, machine learning, application monitoring, and so on. Amazon Web Services was the first IaaS provider and remains the leader, followed by Microsoft Azure, Google Cloud Platform, and IBM Cloud.

PaaS (platform as a service)

PaaS provides sets of services and workflows that specifically target developers, who can use shared tools, processes, and APIs to accelerate the development, testing, and deployment of applications. Salesforce's Heroku and Force.com are popular public cloud PaaS offerings pivotal cloud Foundry and Red Hat's Open Shift can be deployed on premises or accessed through the major public clouds.

FaaS (functions as a service)

FaaS the cloud version of serverless computing, adds another layer of abstraction to PaaS, so that developers are completely insulated from everything in the stack below their code. Instead of futzing with virtual servers, containers, and application runtimes, they upload narrowly functional blocks of code, and set them to be triggered by a certain event (such as a form submission or uploaded file). All the major clouds offer FaaS on top of IaaS: AWS Lambda, Azure Functions, Google Cloud Functions, and IBM open whisk. A special benefit of Faas applications is that they consume no IaaS resources until an event occurs, reducing pay-per-use fees.

The cloud's main appeal is to reduce the time to market of applications that need to scale dynamically. Increasingly, however, developers are drawn to the cloud by the abundance of advanced new services that can be incorporated into applications, from machine learning to internet of things connectivity.

REFERENCES

https://imanagerpublications.com/

https://www.infoworld.com/

https://www.google.com/

LavanyaChamarthi

(19MCA05)

DISTRIBUTED NETWORK ARCHITECTURE

A distributed network architecture is a system that comes with several advantages that can significantly benefit your organisation. A distributed network architecture takes care of three main aspects of systems with large networks. With distributed network architecture a single central control system can be maintained, but the load can be distributed among several local sites. These sites can be physically separated from each other but connected via the internet. And, if one system fails, the others can continue to work without being affected. With single servers, there is always the possibility of an overload as the network grows. Single servers are also prone to the whole network going down if the server is affected. With a distributed system the load is shared among the different systems which makes networking quicker and more efficient. There is no loss of configuration if the central server experiences any problems because it is distributed among the secondary systems. However, the central system also has the advantage of being able to oversee all operations, make security changes, and view the status of the other sites.

Working of Distributed Network Architecture

In a distributed network architecture there are multiple systems that all interact with each other but are also capable of functioning on their own. There is one main system that is connected to the others, and that system can have control over the others. However, each system also has the autonomy to be able to work independently of the other systems of the network. Each node on the system has an administrator, and the tasks that each administrator has control over can be determined by the central node. Each of the sites on the distributed network can vary in size, and some could have thousands of devices connected to the network while some might just have a few. Regardless of the size of each site, all the devices can ultimately be connected back to the centre.

What Are the Benefits of Distributed Network Architecture?

Scalable: A distributed network architecture makes scalability a lot simpler than single networks. Because the **load is distributed**, new devices can be added and configured to the network without much disruption to the whole network.

More efficient: The administrator of the central network can have as much or as little control as required at each time. This administrator can delegate tasks to node administrators and concentrate on other tasks at hand.

More reliable: Because of the way the distributed network architecture is set up the system is more robust and reliable. It uses less bandwidth and can tolerate hardware or network failures much better than a single system. Also, if files on one system get corrupted, it doesn't affect the entire system. With a distributed network architecture in place companies that have networks of a large number of devices, can be more secure and tolerant to setbacks.

The purpose of an automated management solution for security and network devices is to perform various tasks to backup and preserve configurations, Automated Backup and Recovery, Network Task Automation, Inventory Management, Network Visualizations. For organizations with not just a large number of devices, but also devices at different locations, it can be a nightmare to sync the configuration management of every device. If the organization already uses distributed network architecture, then using the system for the management solution works to a great advantage. The automated security and network device management solutions can be uploaded to the central system on the distributed network, and from there it can be sent to the various nodes which in turn distribute it to the devices on the local network. The system helps to reduce both costs and time and makes configuration management more efficient. The distributed network architecture allows the different administrators management access from their nodes. From these nodes, the required operational tasks can be enforced, and status information and backup configuration files can be collected. This type of architecture eliminates the need for multiple firewall rules when backing up remote devices, and it enables data redundancy between the management and agent systems, providing an additional layer of redundancy. Virtual systems can also be added on to the network and synced with the management solution. Every administrator can view the devices assigned to them and manage their own network. This system helps to ease the load of the center especially if there is a large number of devices on the network. Using the management solution on a distributed network architecture ensures that all the devices are covered under the automated tasks.

References:

https://www.hex64.net/what-is-distributed-network-architecture-and-how-does-it-benefit-your-organization/

https://www.tutorialspoint.com/software_architecture_design/distributed_architecture.htm

DEEPIKA N

19MCA06

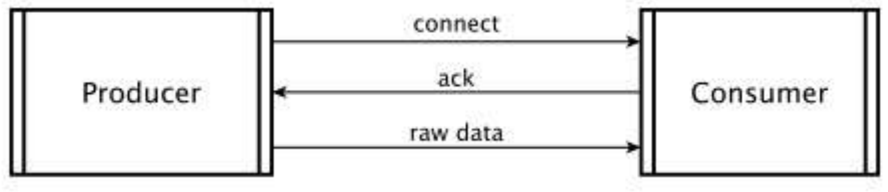
MODERN MESSAGING FOR DISTRIBUTED SYSTEMS

INTRODUCTION:

The modern messaging for distributed is communication and system integration. The messaging features is then provided, followed by the major technologies for messaging, from broker to broker-less systems. The list of successful stories concerning the use of messaging to solve the problem of communication for distributed applications is presented.

Coupled communication Modern distributed systems can be comprised of hundreds, if not thousands, of applications operating in multiple tiers and providing different services and functionality to each other. There are many challenges such as network unreliability, strong coupling of producers and consumers and the heterogeneity of applications which need to be addressed to build a solid and reliable system.

Connection-oriented communication is a solution to exchange information among remote entities. Consider opening a socket over a connection-oriented protocol such as TCP/IP and transmit a raw stream of data through it. That would be a fast and inexpensive way to exchange information, but at the same time that tightly-coupled communication would be based on a number of assumptions which needs to be satisfied in order the communication to take place.



Messaging systems support different communication models, each one defining how the information is exchanged among producer and consumer. The most common communication models are queues and topics. Queue are used to implement a point-to-point communication, where, if no consumer exists when the information is produced, the message is kept in the channel for later delivery, whilst if there are multiple consumers the message is delivered only once. Topic is for the classic publish/subscribe scenario, where if no consumer exists the message is discarded and in case of multiple consumers the message system delivers it to each of them. A part from queue and topic which are widely supported, more complex delivery semantics exist at protocol level and many others are middleware-specific.

Message brokers are solid and reliable technology used as transport layer building blocks in many projects and services, both within the physics community and outside. In the recent years, a new generation of systems is promoting messaging for low-latency / high-throughput / data-intensive communication, as presented , narrowing use cases and relaxing assumptions, but pushing the boundaries of messaging applications towards new domains.

REFERENCE:

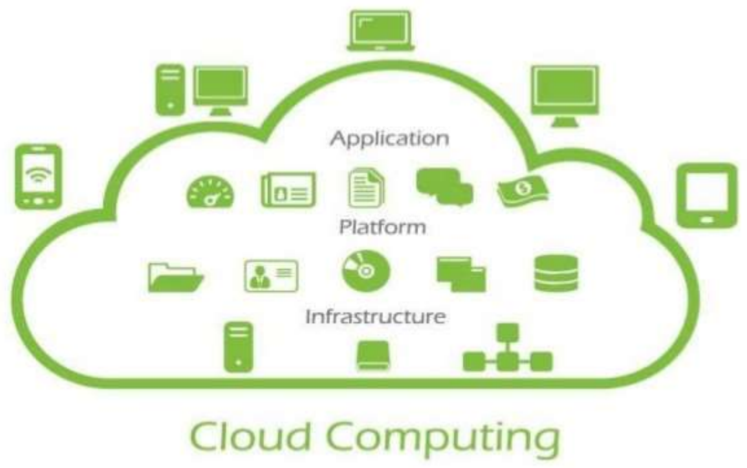
http://iopscience.iop.org/article/10.1088/1742-6596/608/1/012038/pdf

S.JENIFER

19MCA07

GREEN CLOUD COMPUTING

Green Cloud is the study and practice of designing, manufacturing, using a disposing of computers, servers and associated subsystems. Key issues are energy efficiency in computing and promoting environmentally friendly computer technologies. The goal of green cloud computing is to promote the recyclability or biodegradability of defunct products and factory waste by reducing the use of hazardous materials and maximizing the energy efficiency during the product's lifetime.



The objective is to reduce power consumption of the data center. The green cloud architecture is aiming to reduce data center power consumption. The main advantage of the architecture of green cloud computing is that it ensures real-time performance while reducing the energy consumption of the IDC means (internet data center). Migrating to the cloud means fewer machines and less hardware, which translates into lower cooling and space requirements

Examples:

- Purchasing from Environmentally Committed Companies.
- Participating in Electronic Recycling Programs.
- Deploying Virtual Technologies
- Limiting Printing and Recycling Paper.

The goals of green computing are similar to green chemistry: reduce the use of hazardous materials, maximize energy efficiency during the product's lifetime, the recyclability or biodegradability of defunct products and factory waste.

Reference:

https://bigdataanalyticsnews.com/green-cloud-computing-sustainable-use/
https://www.pugetsound.edu/about/offices-services/technology-services/green-computing/

Kavana C (19MCA08)

CONSISTENCY CONTROL FOR DISTRIBUTED INTERACTIVE MEDIA

In this paper we present a generic consistency control service for distributed interactive media, i.e. media which allow a distributed group of users to interact with the medium itself. Consistency control is vital to these media since they typically require that a local copy of the medium's state be maintained by each user's application. Our service helps the applications to keep the local state copies consistent. The main characteristics of this service are as follows: a significant number of inconsistencies are prevented by using a mechanism called local lag. Inconsistencies that cannot be prevented are repaired by an improved time warp algorithm that can be executed locally without burdening the network or the applications of other users. Exceptional situations and consistency during late-join situations are supported by a consistent state request mechanism. Moreover, the service also supports the application in detecting intention conflicts between the actions of distinct users.

The major part of this functionality is based on a media model and the application level protocol for distributed interactive media (RTP/I) and can thus be reused by arbitrary RTP/I-based applications. In order to demonstrate the feasibility of our approach and to evaluate its performance we have integrated the generic consistency service into a shared whiteboard system. Networked computer games, distributed virtual reality systems, and shared whiteboard presentations are prominent examples for a new class of media. Unlike audio and video streaming this media class allows a distributed group of users to interact with the medium itself.

We therefore call this media class "distributed interactive media". This book investigates the distributed interactive media class in detail. It covers a broad range of topics, including an abstract media model, how to ensure consistency, an application-level protocol, and how to develop reusable functionality such as support for late-comers and session recording. The key intention of this book is to demonstrate that distinct distributed interactive media have many problems in common and to show how to solve these problems in a generic and reusable fashion for the whole media class.

References

Edw97.W.K. Edwards. Flexible Conflict Detection and Management in Collaborative Applications. In ACM UIST, pages 139-148, 1997.]]

NISHANTHI

19mca09

MOBILE COMGPUTING

Mobile computing is human–computer interaction in which a computer is expected to be transported during normal usage, which allows for the transmission of data, voice, and video.

Mobile computing involves mobile communication, mobile hardware, and mobile software. Communication issues include ad hoc networks and infrastructure networks as well as communication properties, protocols, data formats, and concrete technologies. Hardware includes mobile devices or device components. Mobile software deals with the characteristics and requirements of mobile applications.

Some of the most common forms of mobile computing devices

Smart cards that can run multiple applications but are typically used for payment, travel, and secure area access.

Mobile phones, telephony devices which can call from a distance through cellular networking technology.

Wearable computers, mostly limited to functional keys and primarily intended as the incorporation of software agents, such as bracelets, key-less implants etc.

Reference:

- https://en.wikipedia.org/wiki/Mobile_computing
- https://india.oup.com/product/mobile-computing-9780199455416

POOJA KUMARI (19MCA10)

DISTRIBUTED DATA ACCESS AND CONTROL

INTRODUCTION:

An important requirement of a distributed data access and control is the ability to support data control, controlling how data is accessed using a high-level language. Data control typically includes view management, access control, and semantic integrity control. Informally, these functions must ensure that authorized users perform correct operations on the database, thus contributing to the maintenance of database integrity. In relational DBMS, data control can be achieved in a uniform fashion. Views, authorizations, and semantic integrity constraints can be defined as rules that the system automatically enforces. The violation of some rules by database operations generally implies the rejection of the effects of some operations (e.g., undoing some updates) or propagating some effects (e.g., updating related data) to preserve the database integrity.

The definition of these rules is part of the administration of the database, a function generally performed by a database administrator (DBA). This person is also in charge of applying the organizational policies. Well-known solutions for data control have been proposed for centralized DBMS. The cost of enforcing data control, which is high in terms of resource utilization in a centralized DBMS, can be prohibitive in a distributed environment.

Since the rules for data control must be stored, the management of a distributed directory is also relevant. The directory of a distributed DBMS can be viewed as a distributed database. There are several ways to store data control definitions, according to the way the directory is managed. Directory information can be stored differently according to its type in other words, some information might be fully replicated, whereas other information might be distributed.

For example, information that is useful at compile time, such as access control information, could be replicated. We emphasize the impact of directory management on the performance of data control mechanisms.

REFERENCES:

Abadi D. J, Carney D, Çetintemel U, Cherniack M, Convey C, Lee S, Stonebraker M, Tatbul N, and Zdonik S. (2003). Aurora: A new model and architecture for data stream management.

Ozsu M.T, Valduriez P. (2011) Data and Access Control. In: Principles of Distributed Database Systems, Third Edition.

PRAJWALA S REDDY

19MCA11

CLOCK SYNCHRONIZATION

INTRODUCTION:

Distributed System is a collection of computers connected via the high speed communication network. In the distributed system, the hardware and software components communicate and coordinate their actions by message passing. Each node in distributed systems can share their resources with other nodes. So, there is need of proper allocation of resources to preserve the state of resources and help coordinate between the several processes. To resolve such conflicts, synchronization is used. Synchronization in distributed systems is achieved via clocks.

External clock synchronization is the one in which an external reference clock is present. It is used as a reference and the nodes in the system can set and adjust their time accordingly.Internal clock synchronization is the one in which each node shares its time with other nodes and all the nodes set and adjust their times accordingly.

There are 2 types of clock synchronization algorithms: Centralized and Distributed.

Centralized is the one in which a time server is used as a reference. The single time server propagates its time to the nodes and all the nodes adjust the time accordingly. It is dependent on single time server so if that node fails, the whole system will lose synchronization. Examples of centralized are- Berkeley Algorithm, Passive Time Server, Active Time Server etc.

Distributed is the one in which there is no centralized time server present. Instead the nodes adjust their time by using their local time and then, taking the average of the differences of time with other nodes. Distributed algorithms overcome the issue of centralized algorithms like the scalability and single point failure. Examples of Distributed algorithms are – Global Averaging Algorithm, Localized Averaging Algorithm, NTP (Network time protocol) etc.

Reasons for Delay in Synchronization:

As discussed above, there are many reasons for a communication delay needs to be minimized to minimize delay and get nearby accurate time. 1944 Neha N. Dalwadi and Dr. Mamta C. Padole

Communication Link Failure: For example, when sending a request message, communication link is working properly and message reaches to the server. If at the time of receiving message, communication link may fail due to some break. And client may not be able to get reply message. After recovery, reply reaches to the client which contains false time value.

Fault Tolerance: During message passing if any component fails, it may cause an inaccurate reading of clock time. So the system should be fault tolerant that can work in the faulty situation and minimize the clock drift value [4].

Propagation Time: Due to heavy traffic or congestion in the network, it may cause large propagation time from server to client. It may cause the inaccurate reading of the clock value in the reply.

Non Receipt of Acknowledgement: It may be possible that due to above reasons client will not get reply within a round trip time and therefore it sends multiple requests to server for synchronization.

The Bandwidth of Communication Link: Due to low bandwidth of communication link, congestion may occur in the network. Therefore request for time will not be able to reach the server or reply message will fail to reach client will affect clock synchronization.

REFERENCES:

https://www.ripublication.com/acst17/acstv10n6_38.pdf

https://www.geeksforgeeks.org/synchronization-in-distributed-systems/

PRIYANKA RANI

19MCA12

ALCHEMI: A .NET-BASED ENTERPRISE GRID COMPUTING SYSTEM

Alchemi1, a .NET based framework that provides the runtime machinery and programming environment required to construct enterprise/desktop grids and develop grid applications. It allows flexible application composition by supporting an object-oriented application programming model in addition to a file-based job model. Cross-platform support is provided via a web services interface and a flexible execution model supports dedicated and non-dedicated (voluntary) execution by grid nodes.

Alchemi was conceived with the aim of making grid construction and development of grid software as easy as possible without sacrificing flexibility, scalability, reliability and extensibility. The key features supported by Alchemi are:

- Internet-based clustering of heterogeneous desktop computers.
- Dedicated or non-dedicated (voluntary) execution by individual nodes.
- Object-oriented grid application programming model (fine-grained abstraction).
- File-based grid job model (coarse-grained abstraction) for grid-enabling legacy applications.
- Web services interface supporting the job model for interoperability with custom grid middleware e.g. for creating a global, cross platform grid environment via a custom resource broker component.

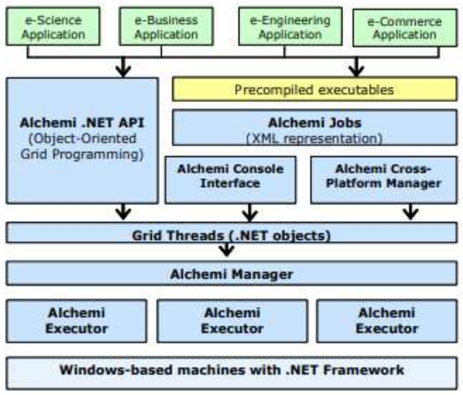

e-Science Application
e-Business Application
e-Engineering Application
e-Commerce Application
Precompiled executables
Alchemi .NET API (Object-Oriented Grid Programming)
Alchemi Jobs (XML representation)
Alchemi Console Interface
Alchemi Cross-Platform Manager
Grid Threads (.NET objects)
Alchemi Manager
Alchemi Executor
Alchemi Executor
Alchemi Executor
Windows-based machines with .NET Framework

Figure 1. A layered architecture for an enterprise grid computing environment.

Reference:

- https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.7986&rep=rep1&type=pdf
- https://www.researchgate.net/publication/1956976_Alchemi_A_NET-based_Grid_Computing_Framework_and_its_Integration_into_Global_Grids

Priyanka S (10MCA13)

PARALLEL PROCESSING

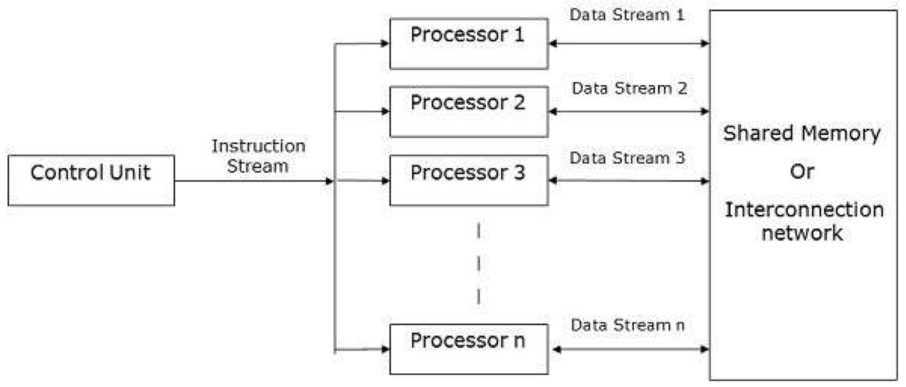
A parallel algorithm can be executed simultaneously on many different processing devices and then combined together to get the correct result. Parallel algorithms are highly useful in processing huge volumes of data in quick time. This tutorial provides an introduction to the design and analysis of parallel algorithms. In addition, it explains the models followed in parallel algorithms, their structures, and implementation.

WHAT IS PARALLELISM?

Parallelism is the process of processing several set of instructions simultaneously. It reduces the total computational time. Parallelism can be implemented by using parallel computers, i.e. a computer with many processors. Parallel computers require parallel algorithm, programming languages, compilers and operating system that support multitasking.In this tutorial, we will discuss only about parallel algorithms. Before moving further, let us first discuss about algorithms and their types.

SIMD (SINGLE INSTRUCTION MULTIPLE DATA) COMPUTERS

SIMD computers contain one control unit, multiple processing units, and shared memory or interconnection network.



Here, one single control unit sends instructions to all processing units. During computation, at each step, all the processors receive a single set of instructions from the control unit and operate on different set of data from the memory unit.

Each of the processing units has its own local memory unit to store both data and instructions. In SIMD computers, processors need to communicate among themselves. This is done by shared memory or by interconnection network.

Reference:

https://www.tutorialspoint.com/parallel_algorithm/parallel_algorithm_introduction.htm#:~:text=A%20parallel%20algorithm%20is%20an,to%20produce%20the%20final%20result.

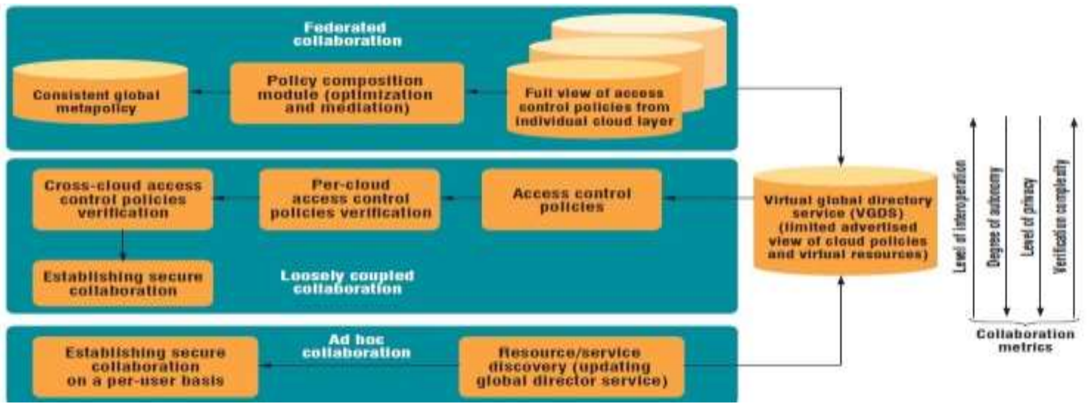
https://www.google.com/search?sxsrf=ALeKk00F3VDFd8Drd9s9cgUAcM9nmTdeZg%3A1616082616802&ei=uHZTYOmoMPSP4-EPgsS60A8&q=parallel+algorithm+in+distributed+systems&oq=parallel++algorithm+in+distributes&gs_lcp=Cgdnd3Mtd2l6EAMYADIGCAAQFhAeOgcIABBHELADOgcIIxCwAhAnOgYIABAHEB46AggAOgcIABCHAhAUOggIIRAWEB0QHlDqMFiFmQFgpr8BaAFwAngCgAHWBIgBxS6SAQw1LjIxLjMuMS4wLjOYAQCgAQGqAQdnd3Mtd2l6yAEIwAEB&sclient=gws-wiz

Roshni rathore

19mca14

DISTRIBUTED APPLICATION CONTROL AND CLOUD INTEGRATION

The growing popularity of cloud computing draws attention to its security challenges, which are particularly exacerbated due to resource sharing. Cloud computing's multitenancy and virtualization features pose unique security and access control challenges due to sharing of physical resources among potential untrusted tenants, resulting in an increased risk of side-channel attacks. Additionally, the interference of multitenancy computation can result in unauthorized information flow. Heterogeneity of services in cloud computing environments demands varying degrees of granularity in access control mechanisms. Therefore, an inadequate or unreliable authorization mechanism can significantly increase the risk of unauthorized use of cloud resources.


Federated collaboration
Consistent global metapolicy
Policy composition module (optimization and mediation)
Full view of access control policies from individual cloud layer
Cross-cloud access control policies verification
Per-cloud access control policies verification
Access control policies
Establishing secure collaboration
Loosely coupled collaboration
Virtual global directory service (VGDS) (limited advertised view of cloud policies and virtual resources)
Level of interoperation
Degree of autonomy
Level of privacy
Verification complexity
Collaboration metrics
Ad hoc collaboration
Establishing secure collaboration on a per-user basis
Resource/service discovery (updating global director service)

- Authorization Requirements
- Multitenancy and Virtualization
- Decentralized Administration
- Secure Distributed Collaboration
- Credential Federation
- Constraint Specification
- Designing a Distributed Cloud Architecture
- Federated Collaboration
- Loosely Coupled Collaboration

Reference

https://www.infoq.com/articles/distributed-access-control-architecture-for-cloud-computing/

SANTHIYA (19MCA16)

PARALLEL COMPUTING

Parallel computing refers to the process of breaking down larger problems into smaller, independent, often similar parts that can be executed simultaneously by multiple processors communicating via shared memory, the results of which are combined upon completion as part of an overall algorithm. .

The primary goal of parallel computing is to increase available computation power for faster application processing and problem solving.

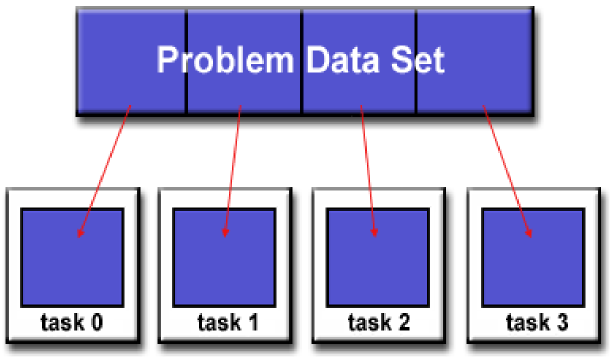
There are generally four types of parallel computing.

Bit-level parallelism: Increases processor word size, which reduces the quantity of instructions the processor must execute in order to perform an operation on variables greater than the length of the word.

Instruction-level parallelism: The hardware approach works upon dynamic parallelism, in which the processor decides at run-time which instructions to execute in parallel; the software approach works upon static parallelism, in which the compiler decides which instructions to execute in parallel

Task parallelism: A form of parallelization of computer code across multiple processors that runs several different tasks at the same time on the same data

Superword-level parallelism: A vectorization technique that can exploit parallelism of inline code.



The importance of parallel computing continues to grow with the increasing usage of multicore processors and GPUs. GPUs work together with CPUs to increase the throughput of data and the number of concurrent calculations within an application. Using the power of parallelism, a GPU can complete more work than a CPU in a given amount of time.

Fundamental of Parallel Computing

Multi-core computing: A multi-core processor is a computer processor integrated circuit with two or more separate processing cores, each of which executes program instructions in parallel.

Symmetric multiprocessing: Shared main memory with full access to all common resources and devices. Each processor has a private cache memory, may be connected using on-chip mesh networks, and can work on any task no matter where the data for that task is located in memory.

Distributed computing: Distributed system components are located on different networked computers that coordinate their actions by communicating via pure HTTP, RPC-like connectors, and message queues.

Massively parallel computing: Refers to the use of numerous computers or computer processors to simultaneously execute a set of computations in parallel.

Reference:

https://www.omnisci.com/technical-glossary/parallel-computing/

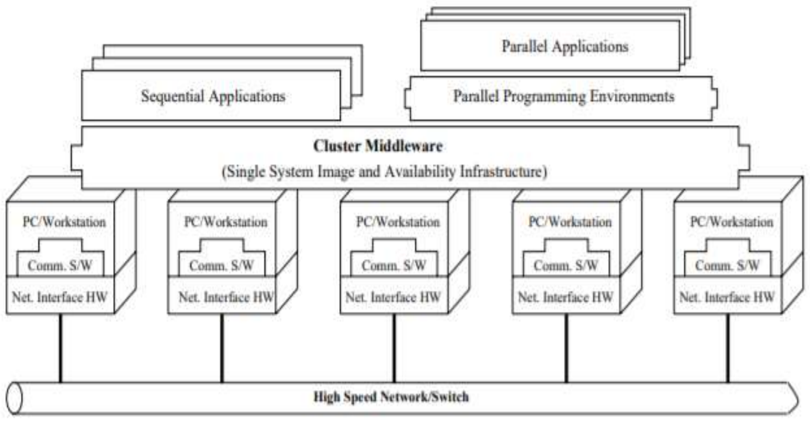
https://www.geeksforgeeks.org/introduction-to-parallel-computing/

A.SARUMATHY

19MCA17

CLUSTER COMPUTING

A cluster is a type of parallel or distributed processing system, which consists of a collection of interconnected stand-alone computers working together as a single, integrated computing resource. A computer node can be a single or multiprocessor system (PCs workstations, or SMPs) with memory, I/O facilities, and an operating system. A cluster generally refers to two or more computers nodes connected together. The nodes can exist in a single cabinet or be physically separated and connected via a LAN an inter connected (LAN-based) cluster of computers can appear as a single system to users and applications. Such a system can provide a cost-effective way to gain features and benefits (fast and reliable services) that have historically been found only on more expensive proprietary shared memory systems the typical architecture of a cluster is shown below.


Parallel Applications
Sequential Applications
Parallel Programming Environments
Cluster Middleware
(Single System Image and Availability Infrastructure)
PC/Workstation
Comm. S/W
Net. Interface HW
High Speed Network/Switch

Clusters Classifications:

Clusters offer the following features at a relatively low cost:

1) High Performance 2) Expandability and Scalability 3) High Throughput 4) High Availability

Cluster technology permits organizations to boost their processing power using standard technology (commodity hardware and software components) that can be acquired/purchased at a relatively low cost. This provides expandability-an affordable upgrade path that lets organizations increase their computing power-while preserving their existing investment and without incurring a lot of extra expenses. The performance of applications also improves with the support of scalable software environment another benefit of clustering is a failover capability that allows a backup computer to take over the tasks of a failed computer located in its cluster.

Reference:

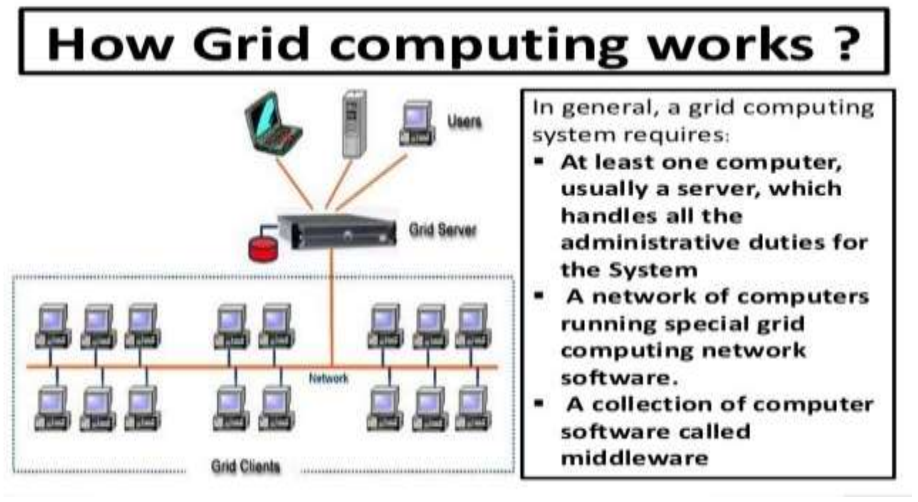
1) https://www.cct.lsu.edu/csc7600/coursemat/Reference/L3/Lecture3_Cluster_Computing_Baker_2.pdf
2) https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.4501&rep=rep1&type=pdf

SUDIPTABALA
(19MCA18)

GRID COMPUTING

Grid Computing can be defined as a network of computers working together to perform a task that would rather be difficult for a single machine. All machines on that network work under the same protocol to act like a virtual supercomputer. The task that they work on may include analyzing huge datasets or simulating situations which require high computing power. Computers on the network contribute resources like processing power and storage capacity to the network.

Grid Computing is a subset of distributed computing, where a virtual super computer comprises of machines on a network connected by some bus, mostly Ethernet or sometimes the Internet. It can also be seen as a form of Parallel Computing where instead of many CPU cores on a single machine, it contains multiple cores spread across various locations. The concept of grid computing isn't new, but it is not yet perfected as there are no standard rules and protocols established and accepted by people.



How Grid computing works ?

In general, a grid computing system requires:

- At least one computer, usually a server, which handles all the administrative duties for the System
- A network of computers running special grid computing network software.
- A collection of computer software called middleware

Advantages of Grid Computing:

It is not centralized, as there are no servers required, except the control node which is just used for controlling and not for processing.

Multiple heterogeneous machines i.e. machines with different Operating Systems can use a single grid computing network.

Tasks can be performed parallel across various physical locations and the users don't have to pay for it with money.

Reference:

https://www.geeksforgeeks.org/grid-computing/

https://www.ias.ac.in/article/fulltext/reso/021/05/0401-0415

SWATI KUMARI (19MCA19)

DATA REPLICATION IN DISTRIBUTED SYSTEMS

Data replication

Data replication is the process in which the data is copied at multiple locations (Different computers or servers) to improve the availability of data.

Types of data replication

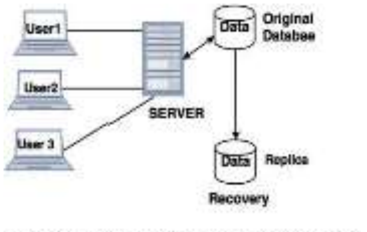
1. Synchronous Replication:
In synchronous replication, the replica will be modified immediately after some changes are made in the relation table.

2. Asynchronous replication:
In asynchronous replication, the replica will be modified after commit is fired on to the database.

Replication Schemes

1. Full Replication

In full replication scheme, the database is available to almost every location or user in communication network.



Full Replication Process In Distributed System

Advantages of full replication

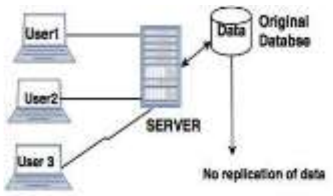
- High availability of data, as database is available to almost every location.
- Faster execution of queries.

Disadvantages of full replication

- Concurrency control is difficult to achieve in full replication.
- Update operation is slower.

2. No Replication

No replication means, each fragment is stored exactly at one location.



No Replication Process in Distributed Databases

Advantages of no replication

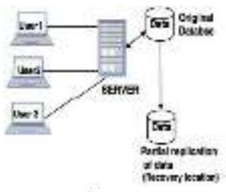
- Concurrency can be minimized.
- Easy recovery of data.

Disadvantages of no replication

- Poor availability of data.
- Slows down the query execution process, as multiple clients are accessing the same server.

3. Partial replication

Partial replication means only some fragments are replicated from the database.



Partial Replication Process in Distributed System

References:

https://www.slideshare.net/KavyaBarnadhyaHazari/replication-in-distributed-systems

https://medium.com/@sandeep4.verma/data-replication-in-distributed-systems-part-1-13f52410faa3

Syeda Saniya Kayeenath

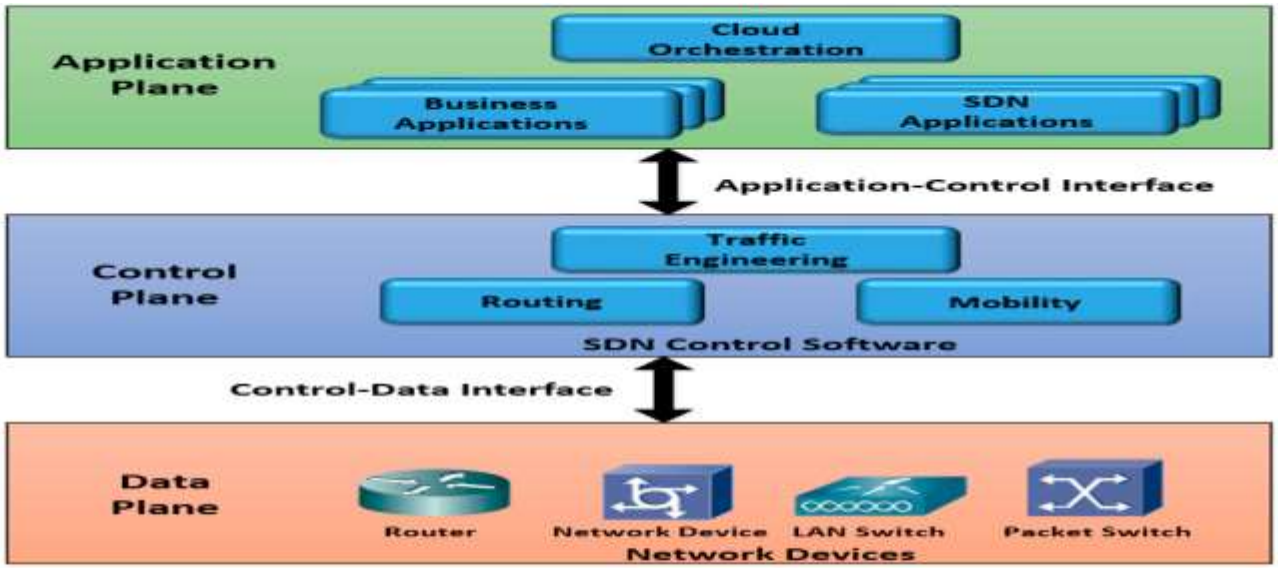
19MCA20

Extensible and Scalable Network Monitoring Using Open SAFE

The Administrators of today's networks are highly interested in monitoring traffic for purposes of collecting statistics, detecting intrusions, and providing forensic evidence.

The Contemporary computer networks have an interesting issue at the same time the dependency we place upon these networks is increasing. The bandwidth is also increasing. With this, the speed of computer networking is out-pacing our ability to effectively monitor them for safety. Ideally network security is best applied with defense in layers at the border, at the administrative boundary and at the edge.

The last of these, the edge, is the only place where reasonable security solutions currently exist for most applications. End-hosts have a diverse array of appropriate solutions to solve their security problems. However, as end-hosts are also very complicated, they cannot be solely trusted for ensuring their safety. Monitoring network traffic at administrative and border-level boundary ingress and egress points are the traditional mainline defenses. Security happens in two phases: active protection and network monitoring. In this work, we are looking at network monitoring. Typically, this monitoring is provided either via middle boxes placed directly on the network path or via inspection of copies of traffic at interesting points in the network.



References:

1. https://dl.acm.org/citation.cfm?id=1863133.1863141

2. https://www.usenix.org/legacy/events/inmwren10/tech/slides/ballard.pdf

3.https://docplayer.net/1103851-Extensible-and-scalable-network-monitoring-using-opensafe.html

P.YOGAPRIYA

(19MCA21)