

Chapter 2

Green Symbiotic Cloud Communications

Abstract Cloud computing, a *TCP/IP* based development, is essentially an integration of computer technologies such as *HPC*, massive memory resource handling, high-speed networks and reliable system architecture. A unified definition of cloud computing doesn't exist with researchers and industrialists globally having listed up to 22 definitions to provide a comprehensive analysis of all the characteristics of Cloud Computing. However Cloud Computing mainly entails as a service that is outsourced, and does not symbolically represent a cloud as we observe in Nature. Classified exhaustively, clouds fit into the following categories—public, private, community and hybrid—however, without much exclusivity. This chapter views the emblem of cloud computing from a different perspective by emulating the geographical cloud as it appears in nature with properties of abstraction and virtualization. The chapter further introduces a first of its kind concept of Cloud Communications. To the best of our knowledge this is an archetype approach of incorporating the communications infrastructure into the cloud. The chapter proposes a Green Symbiotic Cloud (*GSC*) paradigm, which is an amalgamation of all sorts of clouds, with the elimination/minimization of reliance on data-centers, agent-based cooperative approaches and self-managed platforms inherent to systems of the future. Backed by concepts of abstraction and virtualized infrastructure and shared resource pools in one's own local area network, the proposed paradigm offers impetus to revolutionize cloud computing. Taking virtualization to an entirely new level by offering a more local, energy-efficient, synergistic system comprised of individual agents sharing not just resources but knowledge/intelligence in the cloud, it basically emulates the cloud as it appears in nature.

Cloud computing, a *TCP/IP* based development, is essentially an integration of computer technologies such as *HPC*, massive memory resource handling, high-speed networks and reliable system architecture [1–3]. However, its reliability rests on optimality of the connecting protocols, maturity of data-center technologies and service-level agreements. Because a unified definition of cloud computing doesn't exist, proposing a change makes it of utmost importance to lay out essential elements of cloud computing and define it. This will assist us in determining the scope of research and suitability of applications running as cloud services. The key

characteristics of cloud computing are agility, low cost, device and location independence, multi-tenancy, high reliability, high scalability, security and sustainability. Researchers and industrialists globally have listed up to 22 definitions and provide a comprehensive analysis of all the characteristics of Cloud Computing [3–6]. “Clouds are large pools of virtualized resources which are easy to access, secure and reliable. There are 10 characteristics of cloud computing in their sum-up: user-friendliness, scalability, resource optimization, pay-per-use, virtualization, Internet-centric, variety of resources, learning-based adaptation, service *SLAs* (Service-Level Agreements) and infrastructure *SLAs*.” Cloud computing, as a platform, possesses characteristics of both clusters and grids and primarily provides services to users without knowing much about the infrastructure.

Classified exhaustively, clouds fit into the following categories—public, private, community and hybrid—however, without much exclusivity [4, 7]. A private cloud is hosted within an enterprise, behind its firewall, and intended for use inside the enterprise’s offices only; in such cases, a single firm invests in and manages its own cloud infrastructure. It avoids establishment of networks based on large number of low performance systems and instead benefits from the pooling of a smaller number of centrally maintained computing and storage resources [8]. In contrast, a public cloud is hosted on the Internet and designed to be used by any user with an Internet connection to provide a similar range of capabilities and services [8–11]. While they may not differ much under technical considerations, security considerations may be substantially different for services (applications, storage, and other resources) that are made available by a service provider (independent owner and operator) for a public audience and when communication is effected over a non-trusted network.

Community clouds, a method to share infrastructure between several enterprises of a community, stems from the firms’ need to address certain common concerns (security, compliance, jurisdiction, etc.) [10]. Being managed either internally or by a third party, the costs are spread over fewer users than in a public cloud. Hybrid clouds are formed out of the composition of two or more clouds (private, community or public) that remain unique entities bound together and offer the benefits of multiple deployment models [12]. Such composition promotes new synergies and implementation options for cloud services while permitting organizations to use public cloud computing resources only sporadically. This capability enables hybrid clouds to scale elastically. The proposed Green Symbiotic Cloud (*GSC*) paradigm is an amalgamation of all sorts of clouds, with the elimination/minimization of reliance on data-centers. A self-configuring scheme *GSC* Communications proposed in this book treats all networked devices in a locality as the cloud infrastructure.

Cloud services are a discrete set consisting of elements of the sort *XaaS* (‘*X*’ as a service) [1, 4, 7]. For instance, a user can get service from a full computer infrastructure through the Internet. This kind of service is called Infrastructure as a Service (*IaaS*). Internet-based services such as storage and databases are part of the *IaaS*. Other types of services on the Internet are Platform as a Service (*PaaS*) and Software as a Service (*SaaS*). *PaaS* offers full or partial application development that users can access, while *SaaS* provides a complete application, such as Organization resource management, through the Internet. In the typical cloud service model, providers of

IaaS offer physical machines or more often, virtual machines and other resources. A hypervisor runs the virtual machines as guests. A collective of such hypervisors within the cloud's executive support system help support large numbers of virtual machines and offer the ability to scale services up and down according to customers' varying requirements. *IaaS* clouds often offer additional resources such as a virtual machine disk image library, block- and file-based storage, firewalls, load balancers, *IP* addresses, virtual local area networks and software bundles. *IaaS*-cloud providers supply these resources on-demand from their large pools installed in data-centers [6, 13, 14].

In the *PaaS* model, cloud providers deliver a computing platform typically including operating system, programming language execution environment, database and web server. Application developers can develop and run their software solutions on a cloud platform without the cost and complexity of buying and managing the underlying hardware and software layers. With some *PaaS* offers, the underlying computer and storage resources scale automatically to match application demand such that the cloud user does not have to allocate resources manually [12, 15, 16].

In the *SaaS* model, cloud providers install and operate application software in the cloud and cloud users access the software from cloud clients. Cloud users do not manage the cloud infrastructure and platform where the application runs. This eliminates the need to install and run the application on the cloud user's own computers, which simplifies maintenance and accommodates a large number of cloud users that cloud applications support [17]. Cloud applications are different from other applications in their scalability—which can be achieved by cloning tasks onto multiple virtual machines at run-time to meet changing work demand. Load balancers distribute the work over the set of virtual machines. This process is transparent to the cloud user, who sees only a single access point. To accommodate a large number of cloud users, cloud applications can be multitenant, i.e. any machine serves more than one cloud user organization. It is common to refer to special type of cloud-based application software with a similar naming convention: desktop as a service, business process as a service, test environment as a service, communication as a service [6, 7, 10].

A category of cloud services where the capability provided to the cloud service user is to use network/transport connectivity services and/or inter-cloud network connectivity services. *NaaS* involves the optimization of resource allocations by considering network and computing resources as a unified whole. Traditional *NaaS* services include flexible and extended VPN and bandwidth on-demand. *NaaS* concept materialization also includes the provision of a virtual network service by the owners of the network infrastructure to a third party [4, 7, 12, 17].

2.1 Transcending Generic Cloud Computing

In essence, *GSC* as shown in Fig. 2.1 proposes a dynamic computing infrastructure free of the data-centers, agent-based cooperative approaches and self-managed platforms inherent to systems of the future. Backed by virtualized infrastructure and

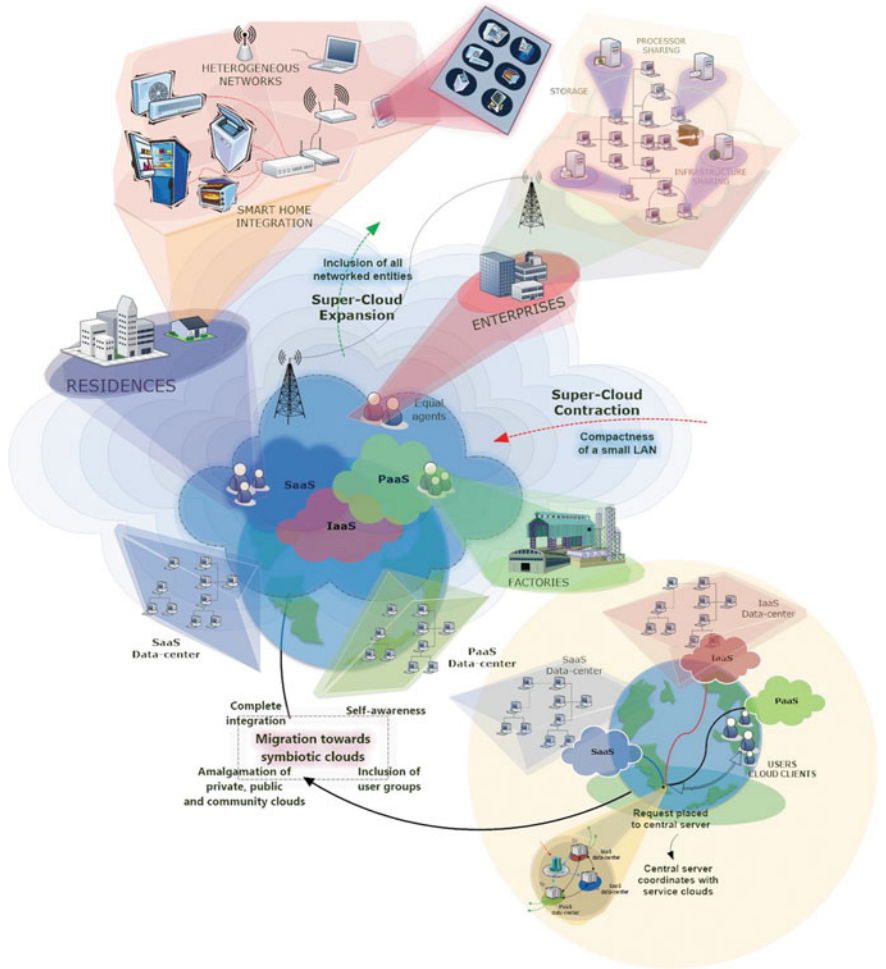


Fig. 2.1 The figure shows the migration from traditional cloud computing/communications architecture to the proposed *GSCC*. Current cloud services rely on dedicated data centres for *IaaS*, *SaaS* and *PaaS*—which may reside at different data centres. Systems of the future will be strongly networked with networks spanning across residences, enterprises, factories, geographies, etc. The figure depicts user groups in different locales being connected symbiotically and maximally utilizing their resources. Smart home systems and other traditionally standalone devices are networked and accessible. The cloud, now a super-cloud, expands and contracts to include/exclude arriving/departing devices accordingly. Users are abstracted and multiple virtualization worlds emerge every instant

shared resource pools in one's own local area network, the proposed paradigm offers impetus to revolutionize cloud computing. Taking virtualization to an entirely new level by offering a more local, energy-efficient, synergistic system comprised of individual agents sharing not just resources but knowledge/intelligence in the cloud, it basically emulates the cloud as it appears in nature.

User experience, a subject of human computer interaction, is an important criterion when evaluating success and user-friendliness of such a paradigm. In cloud computing, user experience improves a lot over ancestors of the likes of grid computing. The core of the experience demands that the services be provided in an inexpensive and easily accessible manner to the cloud user. *GSC* envisions a far more holistic user experience offering cloud services in a more diffused, faster and energy-efficient manner. It offers the user an opportunity to exploit resources in the vicinity rather than relying on a distant data-center. New synergies between both intelligent (with computing power) and unintelligent devices (without computing power) can be explored, promoting progress towards a more integrated super-cloud.

While abstraction is a key component of a consumer-oriented cloud, it needs to make way in a system where there are administrators. If the users are systems administrators and integrators, what they care is how things are maintained in the cloud. They upgrade, install, and virtualize servers and applications. If the users are consumers, they do not care how things are run in the system, rather they look for speed, economy, security and some level of service quality. Grid computing requires the use of software that can divide and farm out pieces of a program as one large system image to a great number of computers. One concern about grids is that if one piece of the software on a node fails, other pieces of the software on other nodes may fail. This is alleviated if that component has a failover component on another node, but problems can still arise if components rely on other pieces of software to accomplish one or more grid computing tasks. Relying on an equal access model on the other hand, *GSC* proposes an advanced level of abstraction for all users. Unless a developer mode is accessed, the diffused cloud and its operatives should remain fully abstracted from the user. Some of these departures are not serious gains, but clouds, like any other ecology, benefit from diversity.

2.2 Pedestals for Systems of Future

2.2.1 Data Caching—*Downloading the WWW onto the Cloud*

Why should a device have to connect to the World Wide Web every time it requires a resource, more so in case of frequently accessed resources? Consider a locally configured cloud network. For instance, all host computers on the local area network at a school may constitute such a network. Notably, most devices connected to such a network sit behind a gateway server and do not possess public *IP* addresses; further,

these devices are configured to fit a standard network topology and are provided *IP* addresses either dynamically or by using a static addressing scheme.

Routinely or not, if a computer *A* were to access a particular webpage *X*, it sends a request that is usually referenced by the *DNS* and forwarded by the gateway, after it updates its *NAT*. Now, if this webpage were recently accessed, referenced and an instance be stored for a fairly large duration, the next time a user requests the same webpage within this duration, the gateway's crawler hops through the *LAN* scavenging all stored instances to locate the desired page. If the page were to be stored in a well-referenced and hierarchical manner, it'd be fairly easy to locate and an appreciable hit-rate can be expected. This model inherently makes the *LAN* cloud more self-reliant. It mitigates the necessity to request and access distant web-servers on the World Wide Web, thereby optimizing the overall throughput. In the process, overall utilization of the abundant local network resources, such as storage and computational capacity, and minimizes load on the *WWW*. Over time, the system is expected to learn from requests, hits and web accesses, and progress towards significantly higher hit-rates.

A very fundamental technique would look like this:

- Heuristic search: release crawlers from both ends (gateway and user equipment). This provides a top-down and a bottom-up radial search. The exponential complexity growth can be fixed by using better, even slightly intelligent crawling algorithms, adapted from web-crawling techniques. A time to live needs to be set adaptively.
- References to high-frequency web resources can be cached on all nodal network elements, viz. routers, bridges, switches and hubs. The terminal link elements possess the densest cache tables while intermediate entities possess only higher frequency results.
- The network is now constantly in a transitory phase updating reference tables at routes. Learning algorithms develop condensed higher priority digests at switches higher up in the hierarchy.

Improvement in performance may be achieved by adapting [16] web-crawling algorithms and [18] search methods to the proposed elastic topology. The proposed scenario is illustrated graphically in Fig. 2.2.

2.2.2 Smart Home Integration

The popularity of home and industrial automation has been increasing in recent years due to much higher affordability and simplicity through smart devices. The provision of remote accessibility has taken automation and security to a whole new level. For instance, a home automation system integrates electrical devices in a house with each other. The techniques employed in home automation include those in building automation as well as the control of domestic activities, such as home entertainment systems, houseplant and yard watering, use of domestic robots, centralized control

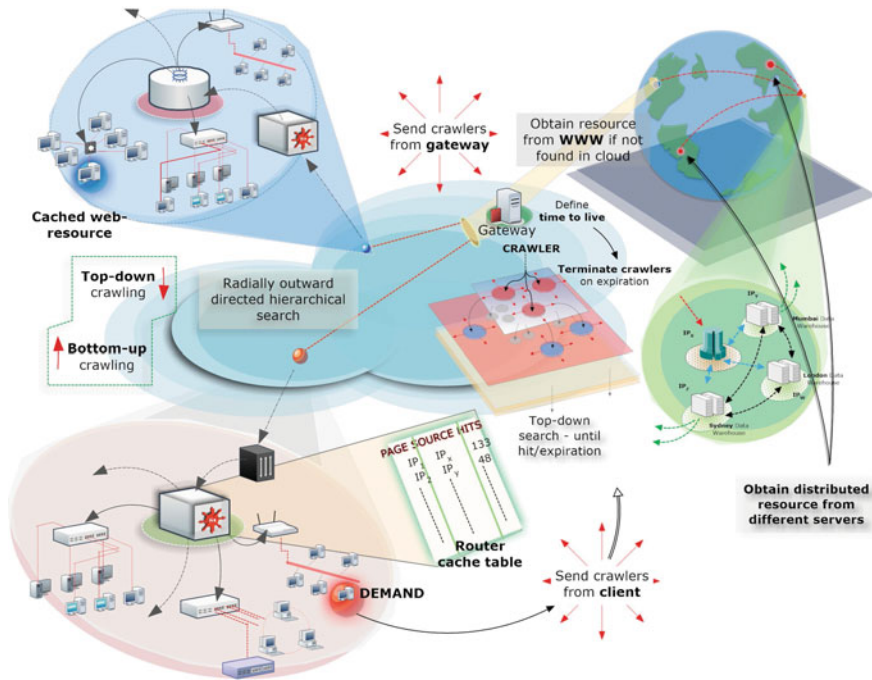


Fig. 2.2 An intelligent data caching scenario based on design postulates of *GSCC* wherein a particular users (*red colour coded*) webpage request is satisfied by another device (*blue colour coded*) in its vicinity, thereby avoiding the *WWW*. It relies on the fact that over time, the network learns the locations popular web-resources cached on the network, and is essentially termed as downloading the *WWW*. A very rudimentary technique has been depicted. **a** **Heuristic search:** release crawlers from both ends (**gateway-top-down** radially outward and **user-equipment-bottom-up** radially inward). The exponential complexity growth can be fixed by using a better, even slightly intelligent crawling algorithm; adapted from web-crawling technique. A time to live needs to be set adaptively; **b** References to high-frequency web resources can be cached on all nodal network elements, viz routers, bridges, switches and hubs. The terminal link elements possess the densest cache tables while intermediate entities possess only higher frequency results; **c** The network is now constantly in a transitory phase updating reference tables at routes. Learning algorithms develop condensed higher priority digests at switches higher up in the hierarchy

of lighting, HVAC (heating, ventilation and air conditioning), appliances, security locks of gates and doors and other systems, to provide improved convenience, comfort, energy efficiency and security. True incorporation into the cloud necessitates that devices be connected through a computer network to allow control by a personal computer/mobile tablet device, and allow remote access through the Internet. Through the integration of information technologies with the home environment, systems and appliances are able to communicate in an integrated manner which results in convenience, energy efficiency and safety benefits, and fosters an intelligent home environment. We consider a case where super-clouds facilitate secure offloading of data logs from smart home systems.

A smart home automation system typically capacitates home owners to monitor and control equipment, utilities and security in a closed space. With a robust sensor network—spanning the entire house/apartment block—monitoring timely changes in equipment performances and surrounding environment, large amounts of data logs are generated and offloaded onto the network. The running footage from safety and security systems—cameras, temperature sensors, sprinkler systems and distributed-locking—impose a stiff load on the storage requirement. Notably, the data stream generated is almost constant in size and more often than not demands large amounts of space for relatively short periods of time. Data that transcends the rest, sporadic in nature, may be stored inexpensively over the local cloud of networked devices rather than on a distant dedicated data storage hub, accessing which repeatedly is cumbersome. Further, relevant data and footage are made available to networked computers and tablet devices; thus home-owners stay updated remotely. The local cloud spanning networked equipment within say, the apartment block, may additionally feed the data logs to the security office for perpetual vigilance. Wavelet-based algorithms providing temporal, spatial and *PNR* scalability are utilized to improve efficiency and accessibility of offloaded data, particularly videos and images [15, 17, 19, 20]. Thus, true smart-home integration can be achieved rather inexpensively sans a service provider or systems maintenance specialist. It can be taken to the extent that even basic lighting and switchboards are automated and intelligence is incorporated therein by a networked system.

2.2.3 Data Security—*Locally Distributed Storage*

While a traditional data-center based cloud leverages economies of scale to optimally store increasing amounts of data, the centralized storage often comes at the price of frequent data-security compromise. Security and threat aversion are critical fields of study within cloud computing, and strong emphasis is being laid on them [3, 21]. However, the community is still far from recognizing a robust and scalable solution to the problem. On the other hand, devices networked through a *GSC* paradigm, in an attempt to increase self-reliance and circumvent usage of data-centers, store data on the local cloud. While this may necessitate the storage of multiple instances of the same data and compromise *QoS* a bit, it nevertheless helps one capitalize on the locally available storage hardware. A consequence, fortunate however, is the fact that this inherently upsurges the security of stored data.

To understand this better, consider a scenario as shown in Fig. 2.3 wherein a 100GB data file needs to be stored on a local cloud. The file is encrypted, broken down into smaller entities and multiple instances of each packet are stored randomly across the local cloud. Based on the confidentiality of data, symmetric key schemes such as *AES 128/192* or weaker schemes such as *DES* may be used for encryption. The encrypted file is split up and distributed across the network. An evaluation of general access and availability trends of devices on the cloud can help decide how many instances of each data packet are to be stored. The encryption scheme, data

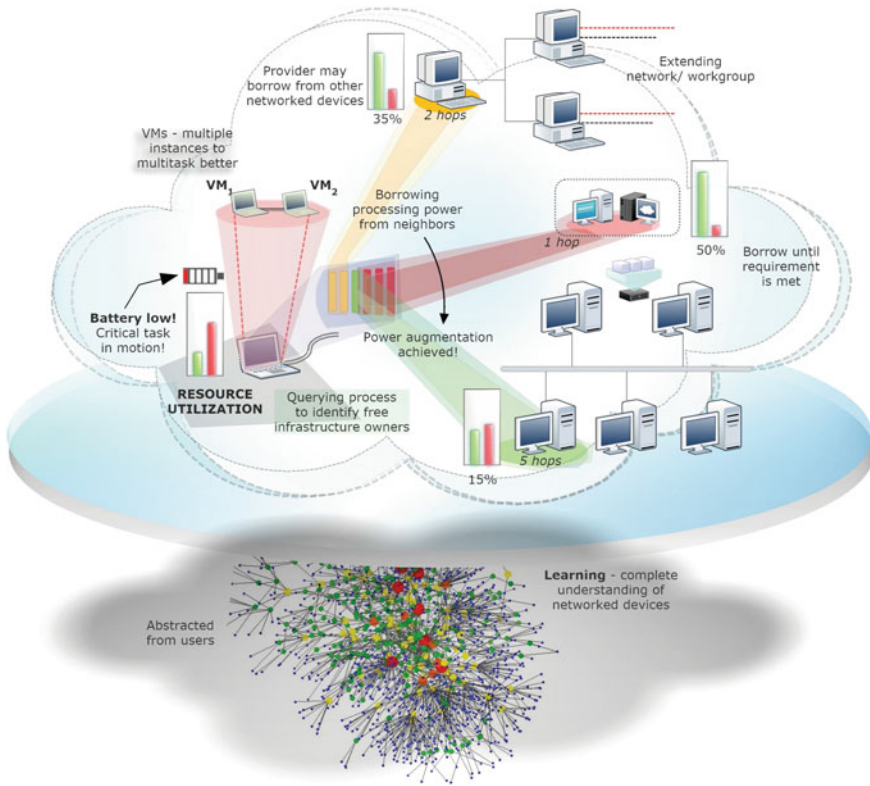


Fig. 2.3 A power augmentation and processor sharing scenario where a user boots into a device and performs certain operations. A significant processing power boost can be achieved by farming out pieces of data and processing them in parallel over co-operative peers on the network. The user is abstracted from the complexity of grid computing and parallel processing and just sees a significant leap in processor speed. An elaborate learning algorithm constantly maps networked devices and acts as a facilitator for services. Further, multiple virtual machines on devices offer parallel computing capacity and deployment of processor sharing. This symbiotic computing power improvement also manifests indirectly as a battery power augmentation, essential for portable devices and generally beneficial

fragmentation and distributed storage collectively enhance the overall security over the existing hardware. Thus, even if few of the hosts on the cloud network were to be compromised, the data remains inaccessible to the attacker and hence secure.

2.2.4 Incorporating Greenness

The energy consumption predicament in information technology equipment has been receiving increasing attention in recent years and there is growing recognition of the

need to manage energy consumption across the entire information and communications technology sector [1, 4–6, 13, 14, 17]. To keep pace with the rapidly growing demand for cloud-services, huge investments are being made towards the development and quick deployment of cloud infrastructure. As a consequence, energy-intensive data-centers are cropping up incessantly around the world. Data-centers and the transmission and switching networks in the Internet account for a significant fraction of total electricity consumption in broadband-enabled countries. In addition to the obvious need to reduce the greenhouse impact of the communications sector, this need to reduce energy consumption is also driven by the engineering challenges and cost of managing the power consumption of large data-centers and associated cooling. Against this, cloud computing will involve increasing size and capacity of data-centers and of networks, but if properly managed, cloud computing can potentially lead to overall energy savings.

While it is important to understand how to minimize energy consumption in data-centers that host cloud computing services, it is also important to consider the energy required to transport data to and from the end-user and the energy consumed by the end-user interface. Studies of energy consumption in cloud computing have focused only on the energy consumed in the data-center [5, 6, 11]. While this accounts for the major chunk of energy consumed, a more comprehensive analysis is required to obtain a clear picture of the total energy consumption of a cloud computing service and understand the potential role of cloud computing to provide energy savings.

2.3 Design Postulates for GSCC Systems

2.3.1 *Virtualization*

A cloud platform can be either virtualized or not. Virtualizing the cloud platform increases the availability of resources and the flexibility of their management (allocation, migration, etc.). It also reduces the cost through hardware multiplexing and helps energy saving. Virtualization is then a key enabling technology of cloud computing. System virtualization refers to the software and hardware techniques that allow partitioning one physical machine into multiple virtual instances that run concurrently and share the underlying physical resources and devices.

2.3.2 *Abstraction*

While abstraction may traditionally refer to concealing details from the user, here, it is used in the sense that users are entitled to a significantly larger chunk of IT resources and computing power than is visible physically. Simply put, this technique will not only abstract the unnecessary technical clutter behind services, but also offer

a huge leap in device performance and user experience. Coupled with learning and growing self-awareness, networked devices of the future are expected to perform at par with small-scale data-centers.

2.3.3 *Distributiveness*

The reliance of cloud-architectures on centralized data-centers, which require constant maintenance, data transportation and conditioning, makes them restrictive in nature. This debilitating characteristic is central even to the web technology wherein web servers are often centralized and remote. Systems of the future ought to promote distributiveness as a leading tenet in approaches and architectures. For instance, clouds could do with departure from the data-center model followed by migration to an agent-based model. In the agent-based model, servers and clients are undefined, rather dynamically defined based on demand and supply of resources within the cloud. At any instance, both servers/service-providers and clients are sporadically distributed. This is expected to yield a significant security improvement in data storage and minimize reliance on data-centers.

2.3.4 *Greenness*

Fragility of the environment and its heightened sensitivity towards emission of pollutants such as carbon-dioxide is pressurizing the scientific community to embrace greenness as a core characteristic in the development of futuristic systems. Systems such as the proposed distributed cloud are expected to reduce energy consumption significantly. Data-centers require immense amounts of brown-energies to sustain 24-h operation and back-forth data transfers around the globe. The data-center model doesn't appear to be very green intuitively, thus departure from the data-center model will inadvertently be a leap towards achieving greenness. To augment energy savings achieved by the new agent-based architecture, systems of the future need to develop and utilize inexpensive green technologies.

2.3.5 *Symbiosis*

Productive interactions between two or more devices, usually over significantly large periods of time, qualify as symbiosis. Symbiosis necessitates the availability of friendly architectures, protocols and perpetual learning. Symbiotic systems are expected to co-exist and co-operate, thus making most of their shared pool of resources. Fairness, acquired through equitable resource allocation and smoothness via increased interaction and self-awareness will improve productivity and

cooperation. A caveat though, is that typical selfish/scavenging-based approaches need to be mitigated.

2.3.6 Pervasiveness/Ubiquity

With the ubiquity of wired and wireless networks, it is not too hard to imagine a perpetual-access cloud anywhere, capable of elastically accommodating incoming and outgoing users. Provisioning almost-omnipresence, a feature often discussed, entails complete self-awareness (knowledge of the system state at any point), aggressive detection of incoming and outgoing users and elastic/friendly protocols. This virtue is expected to expand the super-cloud to all networked devices—portable and static, intelligent and stupid—and propagate ease of access across the cloud. While the data-center model offers better *QoS* and departure from the model poses significant risks of lower *QoS*, development on pervasive technologies only facilitates a more cloud-like on-demand access scenario. A progressive improvement in service quality is expected to transcend a pervasive and adaptive architecture.

2.3.7 Integration

Seamless administration of networks, virtual devices and platforms while elastically accommodating a growing number of devices and tasks into the cloud necessitates smooth integration of systems (smart and otherwise) and exploitation of common ground between networked entities. For instance, developing home automation systems in-sync with evolving cloud standards or more futuristic standards in general ought to be the premise. The integration process should promote accessibility, backward-compatibility, cross-platform movement, seamlessness and concomitant use of multiple platforms among other agenda.

2.3.8 Unification

Ongoing parallel research in IT and hardware, around the globe, is imposing variable methods to achieve similar technological goals. With the perpetration of devices running different technologies, networks using different hardware and protocols/standards and fast-paced evolution symbiosis in clouds necessitates unification of architectures and protocols to some extent. The rapid pace at which technological progress is being achieved, and the visible time-lag between that and adoption of the technologies, demands backward compatibility spanning multiple generations. Unification, achievable through standardization of technology/protocols, improving adaptability in hardware and software and time-tested cognition, has the potential to

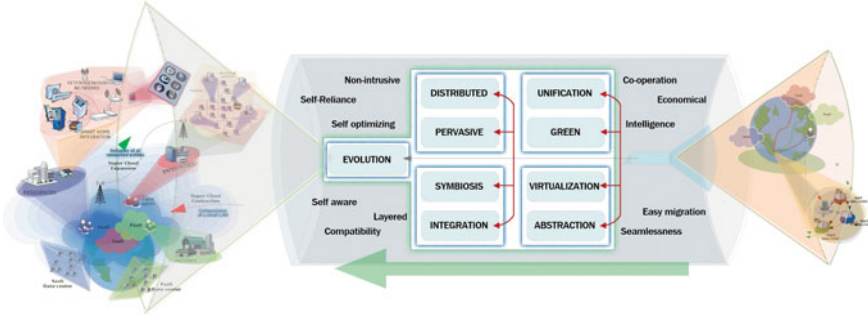


Fig. 2.4 Proposed design postulates for systems of the future

transform the cloud architecture into a more accessible, plug and play model, thus ensuring a multi-fold growth in the user base.

2.3.9 Evolution

Systems of the future need to keep up with the rapid growth in user demands and technological requirements of the age. Persistent migration of users, elastic nature of networks and ever-growing resource pools demand systems to learn constantly and self-optimize resources, addressing, networks, hardware and software. Self-awareness, which transcends time-tested cognition and optimization, may expedite realization of the cloud as a self-reliant, service-providing entity that demands minimal maintenance and user intervention. Not much unlike Darwinian evolution, the evolutionary nature of systems of the future will propagate only the greenest, smoothest, fastest and hopefully fairest architectural features; complete resource recycling and improving *QoS*, greenness and accessibility will serve as the primary guidelines (Fig. 2.4).

2.4 Architectural Design of GSCC Paradigm

This section discusses *GSCC*, a symbiotic and distributed communication paradigm for heterogeneous networks. The green and adaptive approach entails the simultaneous use of multiple communication interfaces, enabling efficient resource utilization. The judiciously designed architecture allows multiple users to access multiple mediums concomitantly, for both uplink and downlink, via virtualized communication ports and Internet Protocol (*IP*) schematic. Principal theoretic feasibility of the hypothesis is established by the linear increase in communication capacity, with minimal energy requirement. Promising simulation and experimental results of a static

scenario are shown, with User Equipment's (*UEs*) connecting to multiple Communication Mediums (*CMs*).

Spectral efficiency and increased throughput in wireless communications, is a topic of paramount importance owing to the substantial increase in user demands [20]. Easy and affordable availability of communication devices like smart mobile phones, laptops, tablets etc., has resulted in its proliferation and subsequent exponential surge in radio traffic [1]. The next quality of service leap is fundamentally expected to come from improvements in network topologies, cooperative communication and virtualization schemes, the amalgamation of cognitive heterogeneous networks and standardization of protocols on such networks [1, 3, 20–22]. Imagine a scenario where a smart phone user makes a video call requiring a capacity of 6 Mbps for uplink and downlink. The user is equipped with unlimited *WLAN* and Long Term Evolution (*LTE*), with each of these mediums providing a capacity of 4 Mbps respectively. In such a case neither of the mediums can provide the required capacity on a standalone basis. Scrutinizing the scenario, one realizes that if simultaneous usage of both the mediums is allowed the desired throughput can be achieved. Ubiquitous access to all available networks thus is the key for the end users to have guaranteed quality of service. The proposed paradigm attempts to integrate this thought evolving a green symbiotic heterogeneous network.

To the best of our knowledge, existing prototypes of heterogeneous networks only concentrate on throughput improvement on the uplink. Communication, however, is a two-way process with the downlink being just as critical as the uplink. In fact, more often than not, downlink carries a larger throughput demand as compared to the uplink [3]. Hence, a significant overall throughput increase necessitates an improvement in both, downlink and uplink data rates. Further, in light of greenness of recent emerging trends and governmental regulations, this too needs to be energy efficient [11, 14]. Consequently prevention of energy drain and complementing the lifetime of battery operated devices like cellular phones, *PDA*s, tablets, etc., forms an essential criterion in the architectural design [11]. As a part of the proposed *GSCC* paradigm we make the following novel contributions in a distributed and hierarchical manner:

1. Proposed is a paradigm, which enables *UEs* to simultaneously utilize all Communications Links (*CLs*) in the vicinity, where it is authorized to.
2. A virtualized communication port and *IP* schematic is proposed, which enables both uplink and downlink communications via all links in a distributed and symbiotic manner.
3. We establish a cognitive decision function and the theoretic networking capacity within the operational constraints and identify the power dynamics of the proposed paradigm.

It is evident that communication devices of the next generation are embedded with multiple radios and technologies. A single smartphone might have the ability to connect to varied *CMs* like cellular *LTE*, *WLAN*, Ethernet, Bluetooth, etc. [2, 5] While concepts like *MIMO*, cognitive radio, etc. enable spatial diversity gain, efficient spectrum utilization and increased *QoS*, their operations are limited to a singular *CM* [18, 19]. Furthermore, currently the limitations exist, where a user is able to only

connect to one *CM* at a particular time. For instance, if a user is connected to *WLAN*, the cellular *LTE* connection for data communication is rendered idle or stopped. The traditional approach prioritizes the available *CMs* based on their *QoS* and picks the best performing one. While it is logical to route majority of the capacity through the strongest medium, the other available mediums, which lie idle, could also be utilized in an energy-efficient way to increase throughput. Alternatively, in an antonymic scenario where high data rate is not required but power efficiency is desired, then only the *CMs* which offer least power-scavenging may be used. Hence, our motivation arises from embedding intelligence in the communication scenario, wherein the *UE* decides cognitively on the usage of *CLs* available to it, for maximizing its throughput and minimizing its operational power dynamics. The approach is coined as Green Symbiotic Cloud Communications (*GSCC*).

Consider a generic communication scenario as depicted in Fig. 2.5a, wherein a user with a smartphone is establishing a call to a user having a laptop. The cellular user has authorization to 4 access points while the laptop user has access to 3 at its end. With the proposed schematic of *GSCC*, both the devices will utilize all the available *CLs* by sending fractions of their data through cognitive splitting to achieve maximum throughput and optimum power usage. This multicast communication is achieved over standard communication protocols like *TCP/IP*, *UDP*, etc., and hence no change in protocol structure is required. In the following subsection, we lay out the basic process flow of realizing *GSCC*.

2.4.1 Process Flow

Consider a generic scenario where UE_a communicates with UE_b , and n' and m' *CLs* are accessible to UE_a and UE_b respectively. Let the *CLs* in the vicinity of UE_a have public IP addresses $\{IP_{1a}, IP_{2a} \dots, IP_{na}\}$ and those in the vicinity of UE_b have $\{IP_{1b}, IP_{2b} \dots, IP_{mb}\}$. For each *CL* that a UE_x is connected to, it is associated with an IP address and port number pair $(^xIP_i, P_i)$. Thus, the set of such identifier pairs being handled by the control layer in each UE_x is $\{(^xIP_1, P_1), (^xIP_2, P_2) \dots, (^xIP_n, P_n)\}$. As represented graphically in Fig. 2.5b, the flow entails 5 major steps:

1. *Scan/Identify*: The *UE*'s scan their environments, using existing provisions on radios/adapters of individual *CMs*. It identifies a list of all the *CLs* that are available and authorized to access.
2. *Connect*: Once all or most *CLs* are identified, the *UE* then establishes connection with all available *CLs*. We necessarily override the *UE*'s preset connection limitations by introducing a software-based control plane layer under the *IP* stack. This control plane comprises of a socket programming code that establishes multiple virtual communication ports. The control plane layer then enables the *UE*'s to handle multiple IP addresses for connectivity through different physical ports of *CMs*.

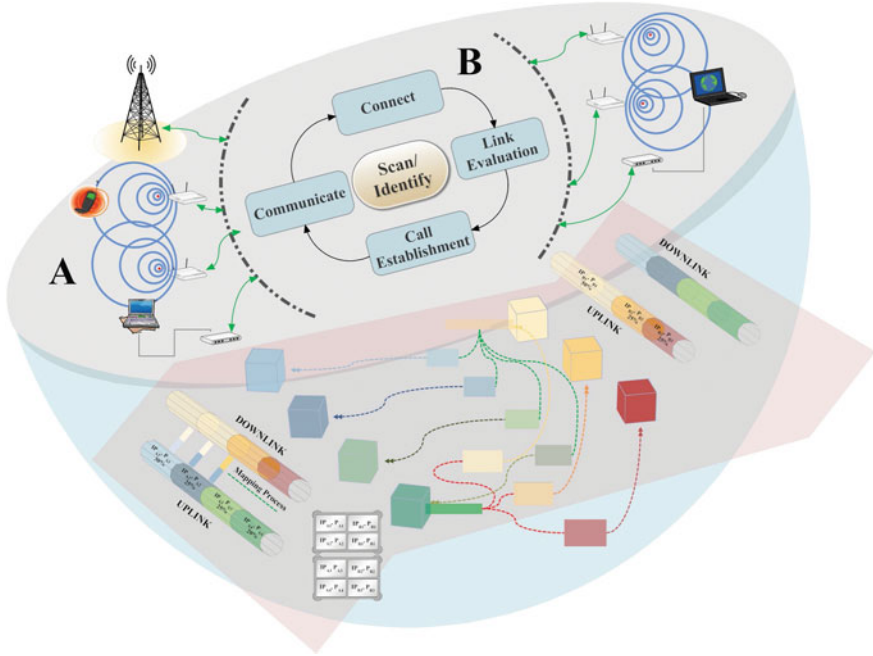


Fig. 2.5 The conceptual outlay of the proposed GSCC is shown where A shows multiple CMs connecting to multiple UEs. We further show a mobile phone connected to 4 CLs establishing a call to a laptop connected to 3 CLs. The connection process flow is outlined in B, which exchanges the virtualized port and IP information of both the UEs. The data flow and mapping for both uplink and downlink is shown in C, where the 4 CLs of UE_a map the data to 3 CLs of UE_b and vice versa. The final IP stack and port hash table is shown in D

3. *Link Evaluation*: The CLs are parametrically evaluated in terms of throughput, BER/data loss, energy consumption per unit data and the associated cost with each CM that the UE encounters. The cognitive splitting of the data is performed based on a decision function, which optimizes the throughput and power consumption.
4. *Communication Establishment*: In this phase, apart from handshaking, peers exchange instantaneous link cost information. This enables the peers to schedule packets on links and label them with appropriate destination IP addresses. Having knowledge of the connectivity situation at both ends allows users to schedule both uplink and downlink via multiple access points.
5. *Communicate*: Once the communication is established, data exchange takes place concomitantly over all the CLs using standard protocols such as *TCP/IP*, *UDP*, etc. The data reception is facilitated via physical ports of the CMs and combined together in the control layer plane of GSCC.

The process continues until a new CL is sighted or an existing one is lost. The new network is seamlessly stitched onto the architecture following the 5-step process flow.

2.4.2 Communication Process Flow

This section illustrates the working mechanism and communication flow of the proposed protocol. Consider a scenario where data transfer is to be initiated by a client to a remote server. Here, a link is considered as a connection established from the server to the client using a specific communication medium. The transport layer implements a cognitive decision function which predetermines parameters such as throughput, congestion, signal-to-noise ratio, power consumption, etc. From the calculated parameters, the transport is split into multiple, virtually parallel transport layers seen in Fig. 2.6 and splits the data packets among the different links with different sessions, each randomly over a predetermined distribution. As shown in Fig. 2.7, the virtual transport block on the server side has a logical one-to-one mapping with the virtual transport block on the client side.

The distribution of the packets transmitted over the network increases the challenge of reassembling at the server side. To counter this challenge, a buffer is introduced on both the server and the client side to store the unsorted data and reassemble the streamlined data segments at the application layer. The total number of links possible is a function whose parameters are the number of network adapters at both the client and the server side. So, the maximum number of logical links possible is $N_s \times N_c$, where N_s and N_c are the number of connections on the server and the client side respectively. This is due to the possibility of a many-to-many logical mapping. However, under the assumptions of a one-to-one mapping, the maximum possible throughput is the linear sum of throughputs of all individual links and maximum power consumption during the transfer of data is the combined power consumption of all links. Figure 2.7 describes the entire process.

With respect to the application, the virtual transport layer performs the functionality of transferring the data from the server to the client or vice versa.

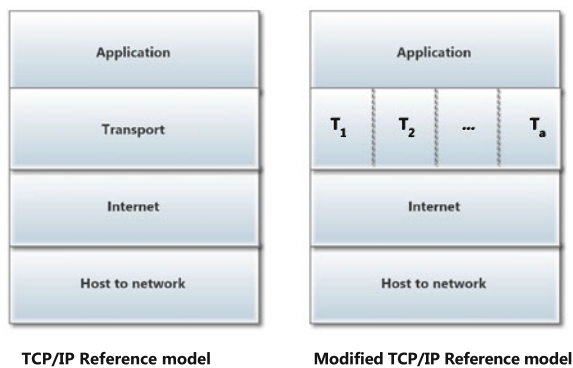


Fig. 2.6 Describes the different *TCP* layers and the modified *TCP* where we virtualize the transport layer to multiple transport layers embedding virtualization in the *GSCC* paradigm

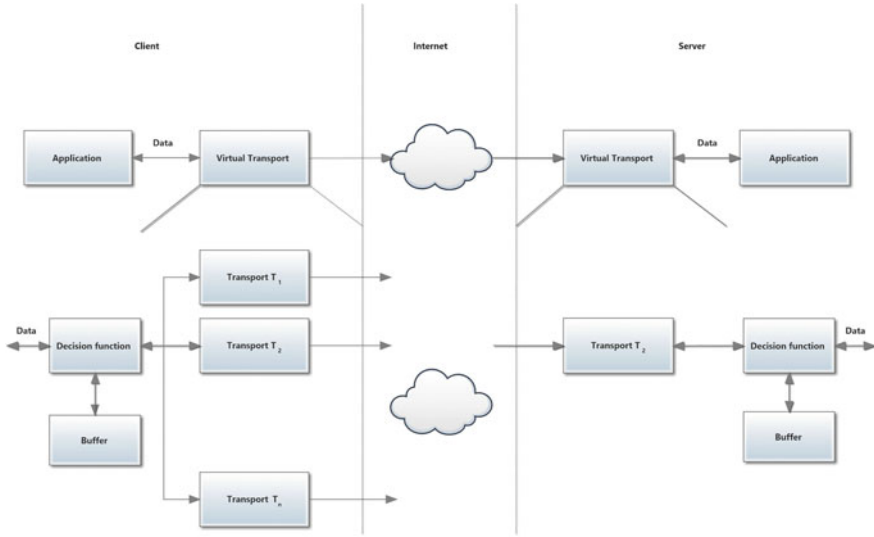


Fig. 2.7 Communication flow in the GSCC paradigm

Decision Function

It has a crucial role in the segmentation and reassembling of the data. The server divides the file into “ n ” different segments and transmission takes place simultaneously. Depending on the decision function, it will split the file in order to optimize total time taken and energy consumption, depending on the user requirements. Optimization of energy and throughputs is done for multiple logical connection scenarios. We classify the networks encountered as follows:

- **Homogeneous network:** When the links are homogeneous, the energy consumed for transferring the data is constant. Therefore, there is no energy factor involved during the splitting.
- **Heterogeneous network:** These links are different as they use different last hops such as *3G*, *Wi-Fi* and Ethernet. For instance, Ethernet usually consumes the least power and has maximum throughput. However, if the user still requires the data at a higher speed than offered by the Ethernet connection, the decision function acts accordingly by considering only maximizing throughput, though the energy consumed is much higher than by utilizing a low-energy medium like Ethernet. But if the user wants the system to run in a power-saving mode, then the decision function disables the *Wi-Fi* and *3G* networks, and uses only Ethernet.

The decision function is dependent on:

1. Capacity of the link $\{T_1, T_2, T_3, \dots, T_n\}$
2. Power consumption of the link $\{P_1, P_2, P_3, \dots, P_n\}$
3. Cost of the link $\{C_1, C_2, C_3, \dots, C_n\}$

If there are ' n ' network adapters for a machine, T_i , P_i and C_i denote capacity, power consumption and the cost of the link respectively for transferring data for the i th link.

Multiple Transport Layers

Data after division will be sent through different transport layers, which correspond to the available different network adapters. The modified *TCP/IP* stack will appear as shown in Fig. 2.6. The original transport layers $\{T_1, T_2, T_3, \dots, T_n\}$ can be used for creating multiple sessions for transferring data in parallel, using the socket application programming interface (*API*). Thus, the virtual transport layers present in the stack are independent of the n different network mediums present, e.g. *Wi-Fi*, *3G/4G*, *Ethernet*, etc. as per the application point of view.

Session Generation

Sessions/connections are created depending on the decision function. Primary and secondary links are created according to the priority order of the connections depending on the decision function outputs. The client initiates a request and the server acknowledges this to initiate a handshake and generate sessions. For a session to be created, client creates a socket which will be used as the unique identity vector C_i containing $\{clientip\ i, clientport\ i, serverip, serverport\}$ where *clientip i* and *clientport i* are the IP address and the port number allotted to the i th network interface respectively. Similarly, we have another vector S on the server which contains $\{clientip\ i, clientport\ i, serverip, serverport\}$. This unique vector is used by the sockets to differentiate different sockets. When client needs to download/upload data, this vector will be used in address headers to identify the client. The data will be sent along with the packet numbers in order to assemble at the other end. The problem occurs when data is sent in multiple sessions, as the data needs to be assembled correctly at the other end. Thus, after dividing the data into packets, they will have a local packet number which will be embedded in the data field of the packet frame and a sessions packet number which will be included in the packet number field of the packet frame, present in the header of the packet. Session packet numbers, which are in sequence, are present in the header of the packet as the firewall will start discarding the packets if continuity in the packet number is not preserved.

The parameters are same as that of the client but an additional parameter status is used to record the current status of the link. During the time a client initiates a request and the server acknowledges it, a handshake is initiated where a list containing the priority order of the links along with the throughputs, cost and power consumption of the links is transferred from the client to the server. For simplicity, assume that the

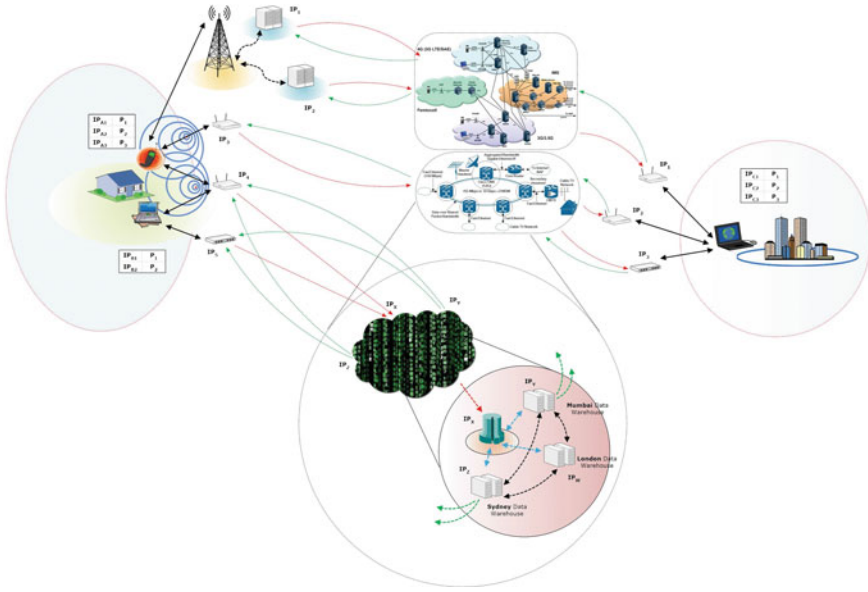


Fig. 2.8 GSCC case scenario: shows a smartphone user in a residential block communicating with a laptop user elsewhere (*right*). The cellphone user simultaneously uses 2 *Wi-Fi* links and one *LTE* connection while the laptop user (*to the right*) uses 2 *Wi-Fi* links and one Ethernet-based broadband connection. An *IP* scheme has been laid out and the uplink and downlink fractions have been displayed next to each link based on a rudimentary calculation. The figure also shows a second laptop user (*left*; residential block) communicating with a distributed data server based in London, Mumbai and Sydney. A similar *IP* scheme has been described for this scenario

file is split uniformly into n segments. Since all n segments are sent simultaneously, the i th segment of the data will travel on the i th link. Packet reassembling is done at the application layer. However, we will require an additional field data sequence in the packet frame which will store the segment number. The connections established are shown in Fig. 2.8.

Creating Multiple Virtual Sessions

In the current scenario, the server stores a session vector for defining the session. If multiple sessions are created from the same client but from a different network adapter (different *IP* address), the server will not be able to recognize whether the link is from the same client or a different one. Creating multiple sessions from the same client with n different network adapters connected can create multiple sessions and use the *GSCC* paradigm for increasing the throughput. Since each network adapter

has a specific *IP* address, the client has a pool of n *IP* addresses. Therefore, while initiating multiple connections to the server, we have to make certain changes to the default *TCP* protocol in order for the server to recognize that the same client is creating multiple sessions. We will define each connection from the client to the server as a communication link. We would require two different kinds of links:

- (a) **Primary link**
- (b) **Secondary link**

The primary link will first initiate a handshake with the server. This link is chosen by the decision function according to the priority order. This link is used for sending information regarding other available links from client, which can be used for establishing the connection. Apart from sending the information, it will act like a normal link sending the packet data.

Other links which are created serve as secondary links, which will be used for sending the packet data. The main functionality of the buffer is storing the data till the layer receives acknowledgement from the other side that the packet is received. If data is not received, re-transmission of the data is done directly from the buffer. It is also used for sorting the received data before sending to the application layer, i.e. if the server is sending/receiving data through multiple links, there will be a delay where some packets get missed. Thus, until the ordered sequence is received, data will be stored in the buffer which will be later assembled and sent to the application layer.

Data Transfer on Virtualized Communication Links

1. **Initialisation:** The decision block initializes by detecting the number of network adapters present on the machine. Decision function calculates the priority of links using the parameters such as throughput of the link, user-defined power consumption and cost of the link.
2. **Prioritizing the links:** The decision block creates a table containing the possible available links to the server which are sorted according to the priority. These are stored in an array using *IP*'s and ports ordered according to the priority.
3. **Creation of link:** The highest priority link will be created first and considered as the primary link. While handshaking, it will discover whether the server is *GSCC*-enabled. If the server is *GSCC*-compatible, remaining connection links will be created according to the priority and are considered as secondary links. Once these connections are established, they are executed in parallel using the socket *API*.
4. **Multiple sessions handling:** The server has a table of *IP* addresses, port numbers, primary session ID's and the status (connected/disconnected) information of each

link. The server needs to know all the *IP* addresses linked up with the client to recognise the connections as arising from the same client. Once the primary link is created, it transfers the priority table to the server. Primary session ID is used as a unique key for identifying the client. Once the secondary connections are established, status indicates connected' in the table. During uplink, the server receives data from all the connections. Using the table, it will assemble the packets in the buffer and send them to the application layer. During downlink the server stores some data in the buffer and sends the data to all the links, which are having connection status as connected' in the table for that particular client.

5. **Closing the link:** Any link can send an '*ack*' to close the session. Closing the session will clear all the *IP* and port addresses attached to the client primary session ID.

Connection Errors

Connection errors are due to connection breakages and are detected when data cannot be transferred through that particular link. We consider the following connection error scenarios:

- **The primary link is broken:** When the primary link is broken, the client can no longer update the server table. So, in order to remove such error, the server chooses the next connected link from the priority table as the primary link. Same is the case at the client side; it will consider the next connected connection as the primary link from the priority table. Connection status of the link is updated to disconnected' and the decision block stops sending data through that link.
- **The secondary link is broken:** If the secondary link is broken, the connection status of the link is changed to disconnected' and the decision block stops sending data through the link.
- **Broken link is re-created:** Client keeps on trying to establish the broken link and if it succeeds in establishing a broken primary link, both client and server assume it again to be the primary link, change its status to connected' and restore the data transmission through it. The existing primary link is again converted to secondary status. When a secondary link establishes connection, the status of the link will be changed from disconnected' to connected' and the data transmission through the link is restored.

The following chapter develops on the architectural base and the concept of the *GSCC* paradigm to establish a theoretical framework validated by experimental results.

Algorithm 1: Client side process

Data: Initializes port numbers and IP addresses of the client

Input the desired location;

while *connection is not established* **do**

 check for host;

if *Host is unreachable* **then**

 initiate random backoff;

 attempt for reconnection;

else

 Create an end communication socket;

 Connect to host using request and wait for acknowledgement;

end

end

Allocate Memory to the buffer and keep application on standby;

while *Check for connectivity* **do**

 initiate data transfer;

 check the status;

if *connection terminated* **then**

 update the server and wait until connection re-established;

else

 check transfer status;

end

end

Algorithm 2: Server side process

Data: Initializes port numbers and IP addresses of the server

while *Listen for incoming connections* **do**

if *Connection using GSCC protocol* **then**

 Accept initialization headers;

 Connect to *GSCC* client;

 Create database containing IP addresses and Port numbers;

 Create end point communication segments;

 initialize buffer and segment the data;

else

 Create an end communication socket;

 Connect to host using request and wait for acknowledgement;

end

end

Select decision criteria;

Commence data transfer;

Check and update the status of the transfer;

Terminate connection when data transfer complete;

References

1. A. Iosup, S. Ostermann, M.N. Yigitbasi, R. Prodan, T. Fahringer, and D. H J Epema. Performance analysis of cloud computing services for many-tasks scientific computing. *Parallel and Distributed Systems, IEEE Transactions on*, 22(6):931–945, 2011.
2. P. Kalagiakos and P. Karampelas. Cloud computing learning. In *Application of Information and Communication Technologies (AICT), 2011 5th International Conference on*, pages 1–4, 2011.
3. M. AbdelBaky, M. Parashar, Hyunjoo Kim, K.E. Jordan, V. Sachdeva, J. Sexton, H. Jamjoom, Zon-Yin Shae, G. Pencheva, R. Tavakoli, and M.F. Wheeler. Enabling high-performance computing as a service. *Computer*, 45(10):72–80, 2012.
4. C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, and A. Monje. On the optimal allocation of virtual resources in cloud computing networks. *Computers, IEEE Transactions on*, 62(6):1060–1071, 2013.
5. B. Kantarci and H.T. Mouftah. Designing an energy-efficient cloud network [invited]. *Optical Communications and Networking, IEEE/OSA Journal of*, 4(11):B101–B113, 2012.
6. N. Xiong, W. Han, and A. vandenBerg. Green cloud computing schemes based on networks: a survey. *Communications, IET*, 6(18):3294–3300, 2012.
7. Frederica Darema. Grid computing and beyond: The context of dynamic data driven applications systems. *Proceedings of the IEEE*, 93(3):692–697, 2005.
8. C.J. Sher Decusatis, A. Carranza, and C.M. DeCusatis. Communication within clouds: open standards and proprietary protocols for data center networking. *Communications Magazine, IEEE*, 50(9):26–33, 2012.
9. Kwang Mong Sim. Agent-based cloud computing. *Services Computing, IEEE Transactions on*, 5(4):564–577, 2012.
10. D. Talia. Clouds meet agents: Toward intelligent cloud services. *Internet Computing, IEEE*, 16(2):78–81, 2012.
11. Jagadish Ghimire and Catherine Rosenberg. Resource allocation, transmission coordination and user association in heterogeneous networks: A flow-based unified approach. *IEEE Transactions on Wireless Communications*, 12(3):11, 2013.
12. M. Bourguiba, K. Haddadou, I. El Korbi, and G. Pujolle. Improving network i/o virtualization for cloud computing. *Parallel and Distributed Systems, IEEE Transactions on*, pp(99), 2013.
13. FeiFei Chen, J.-G. Schneider, Yun Yang, John Grundy, and Qiang He. An energy consumption model and analysis tool for cloud computing environments. *Green and Sustainable Software (GREENS), 2012 First International Workshop on*, pages 45–50, 2012.
14. Renchao Xie, F. Richard Yu, Hong Ji, and Yi Li. Energy-efficient resource allocation for heterogeneous cognitive radio networks with femtocells. *IEEE Transactions on Wireless Communications*, 11(11):10, 2012.
15. M. Shiraz, A. Gani, R. Khokhar, and R. Buyya. A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. *Communications Surveys & Tutorials, IEEE*, PP(99), 2013.
16. Ismail Sengr Altingvde and zgr Ulusoy. Exploiting interclass rules for focused crawling. *Intel-ligent Systems, IEEE*, 19(6):66–73, 2004.
17. M. Lemay, Kim-Khoa Nguyen, B. St. Arnaud, and M. Cheriet. Toward a zero-carbon network: Converging cloud computing and network virtualization. *Internet Computing, IEEE*, 16(6):51–59, 2012.
18. Gaeta R. and Sereno M. Generalized probabilistic flooding in unstructured peer-to-peer networks. *Parallel and Distributed Systems, IEEE Transactions on*, 22(12):2055–2062, 2011.
19. X. Chen, Z. Dai, W. Li, Y. Hu, J. Wu, H. Shi, and S Lu. Prohet: A probabilistic routing protocol with assured delivery rate in wireless heterogeneous sensor networks. *IEEE Transactions on Wireless Communications*, pp(99):8, 2013.

20. Stefan Brueck. Heterogeneous networks in lte-advanced. In *Wireless Communication Systems (ISWCS), 2011 8th International Symposium on*, page 5, 2011.
21. Hsin-Yi Tsai, M. Siebenhaar, A. Miede, Yu-Lun Huang, and R Steinmetz. Threat as a service?: Virtualization's impact on cloud security. *IT Professional*, 14(1):32–37, 2012.
22. D.M. Shila, Yu Cheng, and T. Anjali. Capacity of cooperative wireless networks using multiple channels. In *Communications (ICC), 2010 IEEE International Conference on*, 2010.

Green Symbiotic Cloud Communications

Mustafa, H.D.; Merchant, S.; Desai, U.B.; Baveja, B.M.

2017, XI, 96 p. 51 illus., 46 illus. in color., Hardcover

ISBN: 978-981-10-3511-1