Received June 5, 2019, accepted July 1, 2019, date of publication July 3, 2019, date of current version January 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2926805

Adaptive Capacity Partitioning in Cooperative Computing to Maximize Received Resources

NITIN SINGHA[1], YATINDRA NATH SINGH[2], (Senior Member, IEEE), AND RUCHIR GUPTA[3], (Senior Member, IEEE)

[1]Department of Electronics and Communication Engineering, Indian Institute of Information Technology, Kurnool, Kurnool 518007, India
[2]Department of Electrical Engineering, IIT Kanpur, Kanpur 208016, India
[3]Department of Computer Science and Engineering, IIT (BHU) Varanasi, Varanasi 221005, India

Corresponding author: Nitin Singha (nitin@iiitk.ac.in)

ABSTRACT Users in a cooperative computing environment always have a tendency to free-ride. One can employ incentive mechanisms to prevent such behavior. Some of the cooperative computing scenarios have the same access link shared between upload and download. In such a situation, increasing upload capacity decreases the download capacity and vice versa. Optimal partitioning of link capacity between upload and download needs to be done by each user to maximize its gain (i.e., download) from the network. We model this link capacity partitioning problem as a feedback control system, where feedback (resources received) decides the number of resources to be uploaded by a user. The resulting algorithm called adaptive step size (ASZ) dynamically adjusts the partitioning of link capacity to an optimal value. To compare this approach with others, a metric "level of optimality (U)" is introduced. U achieved by the ASZ is closer to the optimal level than the reputation-based resource allocation policy (existing scheme), thus resulting in its better performance. The ASZ is also integrated with BitTorrent, and the simulation results show that it increases the resources received by the users. The ASZ can provide an efficient solution to the problem of optimal partitioning in real-life distributed networks due to its distributed implementation, robustness to changes in network dynamics, and compatibility with the existing partitioning schemes.

INDEX TERMS Control theory, cooperative communication distributed network, free-riding, reputation system.

I. INTRODUCTION

Many distributed networks use the concept of cooperative computing for their operation. In cooperative computing, members share their resources (e.g., bandwidth, computation power, and storage space) among each other to derive the mutual benefit. Examples of distributed networks employing this notion are file sharing, Domain Name System, BGP-4 routing, application-layer multicast, cloud, mobile and fog computing [1]–[4].

As of now, free-riding was the only major design issue in implementing cooperative computing [1], [5]. Members tend to free-ride, *i.e.*, they do not share/upload resources because there are inherent costs associated in uploading, e.g., cost involved in usage of upload bandwidth. Incentive mechanisms have been proposed in [1], [5], [6], which encourage uploading by allocating resources among members in proportion to the amount they upload. Apart from free-riding, a new challenge of capacity partitioning between upload and download arises in the cooperative computing, due to migration of users from traditional connections like ADSL to the wireless connections. In traditional connections, the upload and download capacities available with the user are fixed, but in most of the wireless networks the capacity allocated for upload and download can be modified by the users [6]–[9]. In networks like WiFi, WLAN, LTE and WiMAX (in time division duplex mode) users are connected to backbone network using an access link. The upload and download data flows through this common access link. The amount of link capacity allocated for the upload and the download can be modified by the user, with their sum being equal to access link capacity. Thus, increasing upload will have a negative effect on download and vice-versa. As in [6], such links are referred as single capacity links. In single capacity links, every user

The associate editor coordinating the review of this manuscript and approving it for publication was Jun Shen.

will strive to optimally partition its link capacity between upload and download, to maximize resources received from the network.

A user cannot download/receive resources greater than its download capacity. Therefore in a bid to receive maximum resources, it will try to allocate its entire link capacity for download. However, incentive mechanisms already being used to prevent free-riding, forces it to upload. In such a scenario, a user will like to fulfill the minimum contribution/upload requirement, to achieve just enough incentive to meet all of its download requirement. The capacity partitioning achieved with this minimum level of upload will be referred as point of optimal partitioning. This is point of optimal partitioning because user receives maximum resources at this point. If a user operates below optimal point, *i.e.*, it reduces it upload then its contribution level decreases, which reduces the resources allocated to it. The other way around, if the user operates above optimal level it has to increase its upload capacity. This will increase its contribution level but its current download capacity will reduce w.r.t. download capacity at optimal partitioning point. This results is reduction in received resources. Hence, at the optimal point a user receives maximum resources.

As optimal point provides maximum resources, a user will strive to operate at this point. However the point of optimal operation may change with time. Thus to continue operation at optimal point, a user needs mechanism which continuously monitor resources received, and accordingly adjust the link capacity partition.

Until now most of users employed traditional connections where capacity allocated for upload and download was fixed. Consequently, not many mechanisms are available to optimally partition link capacity. However with recent advancement in technology, wireless connections can provide high speed data service at lower costs resulting in replacement of traditional connections with wireless connections. The wireless technology is being employed in office, home and hotels to provide Internet services. The focus is shifting towards using wireless technology to provide Internet services in the public places like airports, railway stations, restaurants etc., where people might require uninterrupted Internet connectivity. With widespread use of wireless networks, issue of optimal capacity partitioning becomes significant. Thus in this paper, we propose a mechanism which automatically adjusts capacity partitioning in such a way that it is always close to optimal partitioning point in a distributed setting.

Most of the existing mechanisms [7], [8] are not applicable in distributed scenario, because they require central controlling authority to divide link capacity. Till now for a distributed scenario, Reputation-Based Resource Allocation Policy (RRA) has been proposed in [6]. RRA strives to achieve optimal partitioning by updating the upload bandwidth in fixed step sizes (modifying upload capacity automatically changes download capacity). Under stable condition the upload bandwidth oscillates around the optimum due to fixed sized step. These oscillations reduce the resources received by the users across the network. To minimize these oscillations, we propose to employ a control system.

We model partitioning of total capacity between uplink and downlink, as a feedback control problem. In this control system, the resources received by a user from the network act as a feedback, which decides the system output, *i.e.*, the resources that the user should upload back to the network. The control system seeks to take the system to optimal partitioning level where the bandwidth or capacity received is equal to download capacity available for the current level of upload. Unlike the existing schemes, we employ a PI controller to make size of capacity increment or decrement adaptive such that the step size tends to be 0 as user's sharing level approaches the optimal value. Thus, total capacity partitioning stabilizes around the optimal point and the resource wastage is reduced leading to enhanced efficiency compared to the existing schemes. Main highlights of this paper are listed below.

1) A reputation metric to estimate the contribution level of users is proposed. This metric requires less storage space w.r.t. the existing metrics [6].
2) We model the capacity partitioning mechanism as a control system, where users independently decide their capacity partitioning level. This design is in line with the basic structure of the distributed networks. It is also demonstrated that control system based mechanism is compatible with the existing partitioning schemes. Further, the mechanism is made robust to entry and exit of users from the network by using Proportional Integral (PI) controller.
3) We also propose a metric 'level of optimality(U)', to determine point of optimal partitioning. $U = 1$ signifies that user is optimally partitioning its capacity.
4) The performance comparison of the proposed model with the BitTorrent [10] and the Reputation-Based Resource Allocation Policy (RRA) [6] is presented in the section VII. The control theoretic model of RRA is also derived to show that resource allocation process will be oscillatory, which was subsequently verified during simulation.

The section II discusses the related work. The system model is described in section III, whereas section IV deliberates upon the reputation system and a generic mathematical framework for the resource allocation process. Using this framework, we present the model of the partitioning of access link as a control system in section V. The section VI presents the simulation results, while comparison with the existing capacity partitioning mechanisms is carried out in section VII. Finally section VIII concludes the paper and also discusses the possibilities for the future work.

Notation: A symbol written in calligraphic font represents a set (e.g. $\mathcal{A}$). $\mathbb{R}^+$ represents the interval $[0, \infty)$.

II. RELATED WORK

Several techniques to promote cooperation have been introduced in [1], [5], [10]–[12]. However, they do not provide any mechanism for optimal capacity partitioning. Most of the existing capacity partitioning mechanisms proposed in [7], [8], cannot be used to achieve optimal partitioning in distributed networks due to the following reasons.

1) These mechanisms require a central authority like network administrator to modify the link capacity partitioning according to the current traffic flow. However, a distributed network lacks any central authority.
2) The objective of these mechanisms is not to enhance resources received by optimal partition. Their main aim is to maximize link capacity utilization based upon network traffic, *i.e.*, if currently download requests are more in the network, the above mechanism will increase capacity allocated for download and vice-versa.

In the past many theoretical analyzes have been undertaken to understand the problem of optimal capacity partition. Iosifidis and Koutsopoulos in [13] have modeled total capacity partitioning between upload and download as a utility maximization problem. However, they did not provide any formal framework to utilize the entire link capacity available with the user. Singha and Singh in [14] calculated equilibrium point for capacity partitioning, where no user can enhance its received resources by unilaterally changing its capacity partitioning. Anyway, they did not propose a mechanism to achieve this partitioning equilibrium.

As per our knowledge, there are only two mechanisms which strive to divide the total capacity in a distributed setting optimally. Meo and Milan in [9] contemplated network as market and proved that if users are greedy and employ second price auction, then network reaches an equilibrium stage where the capacity partition converges to a specific value. They also proved that no user in the network could maintain a download rate higher than $\min_{i \in \mathcal{N}} \frac{C_i}{2}$, where $\mathcal{N}$ corresponds to set of all the users, and C_i is the total capacity of the user i. Clearly the slowest link present in the network acts as the bottleneck. Another limitation with this model was that a user can perform only a single upload and download at a given point of time. Limitations in [9] were overcome by Reputation based Resource Allocation (RRA) mechanism proposed by Satsiou and Tassiulasin in [6]. RRA strives to attain the optimal point of resource sharing by dynamically modifying a user's upload capacities in fixed step sizes. Due to fixed step size, total capacity distribution never settles down and keeps on oscillating around the optimal point.

The mechanism presented in this paper tries to alleviate above discussed shortcomings. Using this mechanism, a user can operate more closely to the operating point than the earlier discussed mechanisms. This increases the resources received by a user.

III. NETWORK MODEL

We consider a distributed network of N users, where files are the resources being shared and downloaded by its members. The sharing and downloading of resources contribute to uplink and downlink data flow respectively. Members are connected using single capacity links, where uplink and

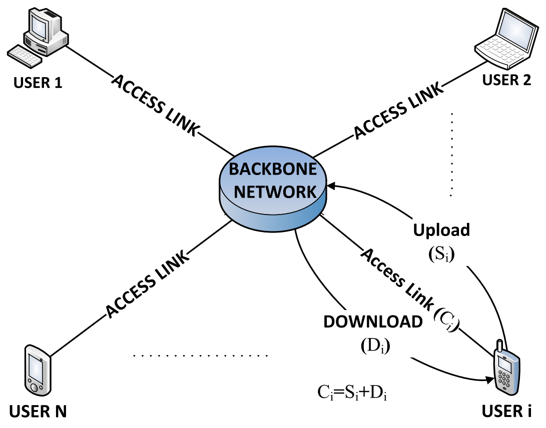


FIGURE 1. Network model.

downlink data of every user flows through the common access link. The access link is a connection, which connects a user to the backbone network (refer Fig. 1). Such connections exist in networks like WiFi, WLAN, LTE and WiMAX (in time division duplex mode). The capacity C_i of access link for any user i is fixed, but the user can modify the division of C_i between upload (S_i) and download (D_i) capacities. Increase in S_i results in decrease in D_i and vice-versa, such that $S_i + D_i = C_i$. The uplink and downlink data utilizes S_i and D_i respectively for their flows. A given data flow cannot be more than its corresponding capacity, *i.e.*, if downlink flow r_i is more than D_i then $r_i - D_i$ resources will get wasted and only D_i amount of resources will be received. If $r_i \leq D_i$, then entire downlink data can be received by the user. The data received by a user i is equal to its utility and is given by

$$utility_i = \min\{r_i, D_i\}. \tag{1}$$

This paper aims to maximize utility, by proposing a partitioning mechanism which optimally divides link capacity. The focus of our analysis is on allocation of the resources among users in the presence of the partitioning mechanism.

It is assumed that the network is fully connected mesh, such that once a user figures out the members from which the resources need to be requested, it directly connects with them. Resource lookout and connection setup mechanisms are out of the scope of this paper. We rather assume that every member in a network has something of interest for every other member. This is in line with the existing resource allocation models used in [6], [9], [14]. Such an assumption is valid for most of the practical distributed networks in use. These networks usually employ techniques like the distributed hash table [15] or flooding [16] for resource lookout. To reduce lookout latency, these techniques make users store data files in demand across the network in their memory. Thus for most of the time, a file to be downloaded is usually available with the other members in the network. This situation can be frequently observed in popular file-sharing applications like BitTorrent. In BitTorrent, a single file is further divided into the large number of chunks [17]. A requester may download the chunks of the same file from different users containing that file. Generally, a requester is interested in downloading more than one file at a given time. Therefore, it needs to download a considerably large number of chunks from the network. Users first download those chunks which are rare in the swarm (The set of users actively uploading and downloading data). This technique called 'rarest first' [10] ensures that each user usually has data chunks which the other downloaders in the swarm desire. A typical swarm consists of a small number of users (around 50) with a requester requiring a large number of chunks and other users containing a diverse variety of chunks due to rarest first technique. All these factors together lead to a situation in BitTorrent, where for most of the network lifetime, a user will always have something of interest for other users across the swarm. Hence a random request in these network will return a chunk of interest. To receive data a requester needs to establish the connection with the other members. A requester i will send g_i requests to the other members for establishing the connection.

After establishing a connection, a requester starts downloads the file from the other user. The user receiving requests for the resources will be referred to as serving user. A serving user j will allocate its resources (S_j) among the requesters as a function of their contribution level. There are different ways to carry out contribution level based resources allocation, e.g., allocating resources in decreasing order of the contribution of the requesters. The actual allocation depends on the resource allocation technique employed by the serving user. The capacity partition mechanism proposed can work with any of the existing allocation techniques [6], [18].

At the end of interaction with a serving user, a requester estimates its contribution level/reputation based upon the resources received. It then updates the reputation of serving user in the global reputation table. Requester also calculates the aggregate utility received. Depending on the utility received, a requester may readjust its link capacity partition, to operate at the optimal partitioning level. This process of downloading and uploading continues for the users until they remains in the network. Using this model, we analyze the resource allocation process in the next section.

IV. RESOURCE ALLOCATION ACROSS P2P NETWORK

The resource allocation process in single capacity links is based on the principle that users will adopt a strategy to derive maximum resource from the network by minimizing their upload. The proposed resource allocation system takes care of this constraint. We expect this resource allocation system to ensure the optimal operation of the system. The resource allocation system can be divided into three components as follows.

1) A mechanism to achieve optimal distribution of total link capacity between upload and download.
2) A mechanism to distribute the upload capacity allocated by the first component among the requesters based on their contribution level.
3) Finally, a reputation metric to estimate the co-operative behavior of various members.

Our main focus is to design a control system which can achieve an optimal partition of the link capacity. To derive transfer function for this control system, we require a mathematical framework of reputation based resource allocation process. Therefore, we first propose a reputation metric, then discuss the resource allocation based upon this metric and derive the transfer function in the next section.

A. REPUTATION METRIC

To make the proposed control system, compatible with the existing reputation metrics, we use a modified version of a common reputation metrics proposed in [6], [12]. Even if in future, somebody proposes altogether a new reputation metric, even then the approach to build the overall control system will remain the same. Thus with the manageable changes, the control system can work with any metric. The used reputation metric is described as follows.

The reputation of user i considers all the transactions where i acts as the resource provider. A transaction is an interaction between two users for the resource download. For a single transaction, the measure of the cooperative behavior is called trust. The trust (t_{ji}) calculated by the requesting user j for the serving user i is defined as the ratio of the bandwidth (r_{ji}) received by the user j to what it has actually demanded (B_{ji}) from the user i during that transaction, *i.e.*,

$$t_{ji} = \frac{r_{ji}}{B_{ji}}. \tag{2}$$

Reputation of the user i at any time instant is calculated using the exponential moving average [19] of old reputation value with the average of current trust values as

$$R_i = (\alpha)R_i^{old} + (1-\alpha)\frac{\sum\limits_{j\in\mathcal{Z}_i} t_{ji}}{|\mathcal{Z}_i|}, \tag{3}$$

where R_i is the reputation of the user i and R_i^{old} is the last reputation value of i available in the network. $\mathcal{Z}_i$ is the set of users which have requested for the resource from the user i after R_i^{old} was calculated. $\mathcal{Z}_i$ is reset every time the reputation R_i of user i is calculated. $|\mathcal{Z}_i|$ is equal to l_i, the number of requests received by i after its last reputation evaluation. We have set $\alpha = \frac{1}{2}$, to assign equal weight to the present as well as past reputation values. This encourages a user to have consistent behavior. If past behavior is not considered, then a user can indulge in free-riding by being non-cooperative when it does not require resources and abruptly become cooperative when it requires resources. The reputation value for $\alpha = \frac{1}{2}$ becomes

$$R_i = \frac{R_i^{old} + \frac{\sum\limits_{j\in\mathcal{Z}_i}\frac{r_{ji}}{B_{ji}}}{l_i}}{2}. \tag{4}$$

This reputation value will be stored in the global reputation table, which is available to every user in the network.

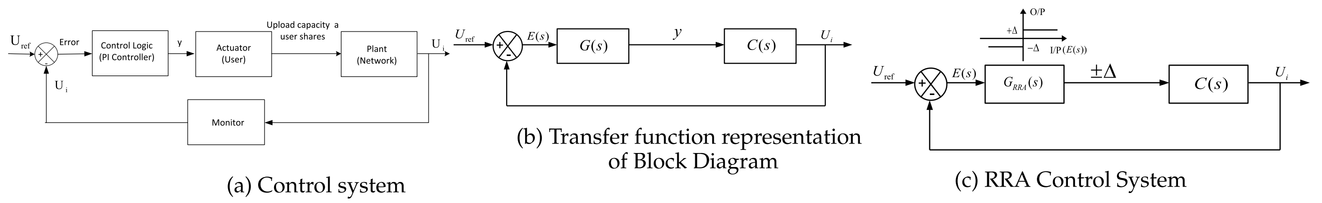


FIGURE 2. **Block Diagram of Control System at user *i*.**

The reputation aggregation (for calculating global reputation) can be achieved using gossiping [20] which has very low complexity in terms of memory, space, and time. Hence without much overhead global reputation can be maintained by the members in the network. We now derive a generic resource distribution mechanism based upon the reputation.

B. REPUTATION BASED RESOURCE DISTRIBUTION AMONG REQUESTERS

To design a control system independent of an underlying reputation based resource distribution technique, we present a generalized mathematical analysis for the resource distribution. This generic framework estimates the amount of bandwidth available to a user from the network. The bandwidth allocated to any requesting user i from j is proportional to its reputation (R_i), bandwidth demanded (B_{ij}) from j and a proportionality constant $k_{ij_{ovd}}$ which caters for change in received bandwidth due to overloading[1] at j. Thus the bandwidth received by i from j is given as

$$r_{ij} = k_{ij_{ovd}} \times R_i \times B_{ij}. \tag{5}$$

The value of $k_{ij_{ovd}}$ lies in the interval $\left[0, \frac{1}{R_i}\right]$. The $k_{ij_{ovd}} = 0$ denotes the worst case of overloading at serving user j, where the receiving user i will not receive any bandwidth. When $k_{ij_{ovd}} = \frac{1}{R_i}$, the bandwidth requirement of user i will be completely satisfied by the user j. The value of proportionality constant $k_{ij_{ovd}}$ varies in each period depending upon the degree of overloading at the serving user. At the same time, for another requester say x requesting the same serving user, the value of $k_{xj_{ovd}}$ will vary depending upon its reputation. The mechanism to estimate $k_{ij_{ovd}}$ is out of the scope of this paper. We just need to ensure that the control system remains stable, even for the maximum possible value of $k_{ij_{ovd}}$.

The total bandwidth (T_i) allocated to a user i for the transactions after the estimate of R_i is

$$T_i = R_i \times \sum_{j \in \mathcal{A}_i} k_{ij_{ovd}} B_{ij}, \tag{6}$$

where $\mathcal{A}_i$ is the set of the users from which i has demanded the resources.

In the subsequent section, we utilize the total bandwidth (T_i) and the reputation (R_i) to derive a control system model at any user i, which optimally divides total capacity between upload and download.

V. MODELING OF LINK CAPACITY PARTITION AS A CONTROL SYSTEM

The partitioning of link capacity between upload and download at the user can be modeled as a classical feedback control system (refer Fig. 2a). This system follows a reference point, which is the optimal point of capacity partitioning, whereas feedback is the ratio of resources allocated to what is demanded by the user. Based upon the feedback, control system dynamically adjusts the total capacity partitioning in such a way that the system starts operating at the optimal point.

The following subsection propose a metric 'level of optimality' (U) for determining the reference point of the control system.

A. REFERENCE OR SET POINT IN CONTROL SYSTEM

1) LEVEL OF OPTIMALITY (U)

The *level of optimality* (U), signifies the resources received by a user in proportion to what it can actually utilize. For a user i, level of optimality is defined as

$$U_i = \frac{T_i}{D_i}, \tag{7}$$

where T_i is the total data rate given by all the users that are currently serving the requester i, and D_i is the current download capacity of i.

At the outset, U may resemble the popular concept of utility (refer (1)) in the computer networks, although it is quite different. The utility corresponds to the total resources received by a user from the network, whereas U is dependent upon the total resources allocated to the user from the network. Sometimes, the user's capacity may not be sufficient to utilize all the allocated resources, therefore these extra resources which are considered in U cannot be received by the user and therefore do not contribute towards the utility of the user. For example, if the user i is offered bandwidth T_i such that $T_i > D_i$. This scenario is possible because users usually demand more[2] [6] than their download capacity to increase their chances of getting the resources from the network. The user i can only utilize the bandwidth equal to its current download capacity D_i and the rest of the

[1]Overloading represents the situation when overall bandwidth demanded by all the requesters from the serving user j is more than what j has shared.

[2]Demanding in excess is usually beneficial when network is overloaded with requests.

offered bandwidth ($T_i - D_i$) gets wasted. The U_i takes $T_i - D_i$ into account, whereas utility does not. We now try to determine the value of U_i which will be optimal point of operation-where a user receives maximum utility.

2) REFERENCE POINT DETERMINATION

A control system always strives to bring the system to the reference point. We set the reference point of the system equal to the point where a user receives maximum utility- maximum resources from the network at the minimum upload. To determine the reference point in terms of 'level of optimality (U_i)', we have divided the feasible region $\mathbb{R}^+$ of variable U_i into three subsets $\mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3$ as follows.

$$U_i \in \begin{cases} \mathcal{U}_1 = [0, 1), & \text{when } T_i < D_i, \\ \mathcal{U}_2 = \{1\}, & \text{when } T_i = D_i, \\ \mathcal{U}_3 = (1, \infty), & \text{when } T_i > D_i. \end{cases}$$

1) ($U_i \in \mathcal{U}_1$): During this state, $T_i < D_i$, therefore entire T_i can be received by the user. Therefore, both utility and U_i derived by the user i from the network increases with increase in T_i. As user always has scope of increasing its utility, so this region does not contain point of maximum satisfaction level. Hence, reference point doesn't lie in set $\mathcal{U}_1$.
2) ($U_i \in \mathcal{U}_3$): Increasing upload capacity (S_i) in this region, results in allocation of more resources (T_i) from the network. However, increased T_i will not be beneficial for the user because it exceeds the maximum data rate D_i a user can handle. Thus when $U_i \in \mathcal{U}_3$, with increase in S_i, U_i increases but the utility decreases. The decrease in utility occurs because D_i decreases due to increase in S_i, which further reduces resources received. Consequently, user i is at loss when its U_i increases beyond 1. Hence, $\mathcal{U}_3$ will not contain the reference point.
3) ($U_i \in \mathcal{U}_2$): This point corresponds to minimum S_i with which a user can be allocated resources equal to its download capacity, *i.e.*, $T_i = D_i$. Minimum S_i implies maximum D_i. Therefore, maximum resources are received at this point by minimum upload. Hence $U_i = 1$ corresponds to the optimal point (point of maximum utility) and serves as the reference point for the control system.

B. CONTROL SYSTEM COMPONENTS

Fig. 2a presents the control system model. The different components of this model are described blow.

1) CONTROLLER

It is used to stabilize the system output to the reference point. Based upon the difference between the reference point (U_{ref}) and the current level of optimality (U), the controller drives the actuator to regulate the output. Generally, PID controller is preferred as controller [21]. In PID, 'P' implies the proportional action which helps a user to reach the steady state faster. A user i has reached the steady state, when it is operating at optimal point, *i.e.*, $U_i = U_{ref}$. 'I' implies the integral action, which reduces the steady state error. Steady state error is the perturbations arising in the system after it has reached the steady state. 'D' implies the derivative action, which is required to counter sudden changes in the output. In the current system differential action is not needed, because once the network reaches the steady state, the output does not change abruptly. Most of the users during network's steady state are operating at optimal partitioning point, and they cannot enhance their utility by changing their upload capacity. As users adhere to their upload capacity, so total shared resources across the network do not change suddenly and neither does output (U) as a consequence. Hence, the PI controller with the transfer function

$$G(s) = K_p \left(1 + \frac{K_i}{s}\right), \tag{8}$$

is sufficient to model the link capacity partitioning process. K_p and K_i are the proportional and integral gain respectively of the PI controller. The values of these gain parameters are calculated in section V-E.

2) ACTUATOR

The role of the actuator [21] in the control loop is to update physical entity based upon the controller output (y). In the system under consideration, the actuator changes the upload capacity of the user by y units in the next period.

3) PLANT

The plant represents the distributed network, which decides the amount of the resources allocated to requesters on the basis of their reputation. The output of the plant pertaining to a user i is the level of optimality U_i, received by i from the network.

4) MONITOR

The function of the monitor is to sense the output and convert it into a form which can be compared with the reference point. In the current system (refer Fig.2a), as the output (U_i) can be directly compared with reference point (U_{ref}), so both input and output from the monitor are U_i. Thus the gain of the monitor is 1.

C. ACHIEVING REFERENCE LEVEL

The monitor in the loop (see Fig. 2a) will sense U_i. U_i will be then compared with the reference point ($U_{ref} = 1$), and the error ($U_{ref} - U_i$) will be fed to the controller. The output of the controller drives the actuator. The actuator modifies the upload capacity which subsequently alters the user's reputation, thereby changing the resources allocated to it. All these changes are directed in the direction so that the U_i received by the user i becomes equal to the U_{ref}. This results in a user always working close to the optimal point.

To complete the design of the proposed control system, we derive the transfer function of the various components in the subsequent subsection.

D. THE CONTROLLED SOFTWARE SYSTEM MODEL

In Fig. 2a, let $C(s)$ denote the transfer function of controlled software system (which includes actuator and Plant) and $G(s)$ represents the transfer function of the controller. Representing various blocks in Fig. 2a with their corresponding transfer functions, we obtain Fig. 2b.

We know derive the transfer function for controlled software system, *i.e.*, $C(s)$. Its input is y (the output of controller) and the output is level of optimality (U). To evaluate $C(s)$, we need to first calculate the output (U).

1) LEVEL OF OPTIMALITY (U_I) RECEIVED BY THE USER *I*

Reputation of user i determines its U_i. Equation (4) gives the reputation of the user i as

$$R_i = \frac{R_i^{old} + \frac{\sum_{j\in\mathcal{Z}_i} \frac{r_{ji}}{B_{ji}}}{l_i}}{2}, \tag{9}$$

where R_i^{old} is the last reputation value of the user available in the network. r_{ji} and B_{ji} are the bandwidth received and demanded respectively by the requester j from i. $\mathcal{Z}_i$ denotes the set of users requesting service from i while l_i is the number of requests catered by i. The total bandwidth (T_i) allocated to the i using (6) is

$$T_i = R_i \times \sum_{k\in\mathcal{A}_i} k_{ik_{ovd}} \times B_{ik} = \frac{R_i^{old} + \frac{\sum_{j\in\mathcal{Z}_i} \frac{r_{ji}}{B_{ji}}}{l_i}}{2} \times \sum_{k\in\mathcal{A}_i} k_{ik_{ovd}} \times B_{ik}, \tag{10}$$

where $k_{ik_{ovd}}$ is a constant which take into account overloading at the serving user k. The serving users allocate resources in proportion to the demands of the requesters. Normally a requester should request according to its download capacity, but it may receive much lesser resources than what it had requested. This may happen because of the overloading of the serving user with a large number of requests. A serving user may not have enough upload capacity to cater to all the requests. In the worst case of overloading, a serving user may be unable to provide any resource to some of the requesters. To optimize for themselves, requesters may follow greedy strategy [6] where they demand more than the download capacity. Therefore, i will place a bandwidth request for the worst case, where only one serving user is reciprocating towards its request. In such scenario, i will place request equal to its current download capacity (D_i), *i.e.*, $B_{ik} = D_i$ and equation (10) becomes

$$T_i = \frac{R_i^{old} + \frac{\sum_{j\in\mathcal{Z}_i} \frac{r_{ji}}{B_{ji}}}{l_i}}{2} \times D_i \sum_{k\in\mathcal{A}_i} k_{ik_{ovd}}. \tag{11}$$

To overcome the problem of requester demanding more than its feasible capacity, the serving user estimates the feasible capacity of the link with the requester [6]. Feasible capacity is the minimum capacity required to support the data rate feasible across the link. Serving user estimates the feasible capacity from the rate of acknowledgments of the TCP flows. Suppose a serving user j sends message packets at the rate of 8 Mb/s to the requester i. However, if the download capacity of link with i is 2 Mb/s, then j will receive the acknowledgment packets at the rate of 2 Mb/s. Thus j decrease the capacity allocated to i accordingly, such that the data rate of 2 Mb/s can be supported. To counter the requests from requesters which are more than their feasible capacity, serving user allocates resources corresponding to the feasible capacity instead of the capacity demanded by the requesters.

As a given link cannot support data rate greater than the feasible capacity and requesters are always demanding more than the feasible capacity, so we assume use of feasible capacity for the reputation evaluation by the requester instead of the capacity demanded. If B_{ji}^{fes} is the feasible capacity of the link between user i and j, then (11) becomes

$$T_i = \frac{R_i^{old} + \frac{\sum_{j\in\mathcal{Z}_i} \frac{r_{ji}}{B_{ji}^{fes}}}{l_i}}{2} \times D_i \sum_{k\in\mathcal{A}_i} k_{ik_{ovd}}. \tag{12}$$

Putting value of T_i in (7), level of optimality received by the user i is

$$U_i = \frac{T_i}{D_i} = \frac{R_i^{old} + \frac{\sum_{j\in\mathcal{Z}_i} \frac{r_{ji}}{B_{ji}^{fes}}}{l_i}}{2} \times \sum_{k\in\mathcal{A}_i} k_{ik_{ovd}}. \tag{13}$$

In the next subsection we use the value of U_i to derive the transfer function $C(s)$.

2) TRANSFER FUNCTION OF CONTROLLED SOFTWARE SYSTEM

For ease of analysis in evaluating $C(s)$, we linearize the non-linear model[3] of the process which optimally partitions total capacity between uplink and downlink. Any non-linear system can be linearized if it works under narrow operating range [21]. This condition is also true for the proposed system where every user for maximum duration, operates around the optimal point *i.e.* $U = U_{ref}$. This happens due to the following reasons:

1) A new entrant can swiftly achieve $U = U_{ref}$ due to the proportional action of the controller.
2) Once $U = U_{ref}$ is achieved, the integral action maintains U's value close to U_{ref} during perturbations arising due to change in network dynamics. For details refer to section VI-C.
3) Later in section VI-B we prove that it is more profitable for the new entrants to initially share their total link

[3]Many complex nonlinear systems like web servers, servomotor, tachometer, synchros have been studied using linearization.

capacity equally between upload and download. This approach also helps new entrants to operate near U_{ref} as soon as they enter the network.

In addition to the above stated arguments, we further substantiate the accuracy of the linearized model using simulation results in section VI-B. Results demonstrate that the linearized version of the proposed system closely follows the output of actual nonlinear system. Hence, the proposed system can be modeled as a linear system. We now derive the transfer functions $C(s)$.

As discussed later in section VI-B, capacity partitioning already starts near the optimal point, so we can neglect the process dynamics[4] and $C(s)$ can be modeled as the static gain c. On linearizing [21], this gain, c can be obtained as the derivative of the output (U) with respect to the input (y). Using (13), the gain c^i corresponding to user i is given by

$$c^i = \frac{dU_i}{dy} = \frac{d\left[\left(\frac{R_i^{old} + \frac{\sum_{j\in\mathcal{Z}_i} \frac{r_{ji}}{B_{ji}}}{l_i}}{2}\right) \times \sum_{k\in\mathcal{A}_i} k_{ik_{ovd}}\right]}{dy}. \quad (14)$$

The c^i is also called the process gain and is equal to the product of gain of actuator and plant (*i.e.*, $C(s)$) with the gain of monitor (*i.e.*, $H(s)$). However as monitor gain is unity, c^i corresponds to $C(s)$. c^i signifies system's sensitivity and is defined as relative distance process variable (U) travels in response to change in the controller output (Y). It is used to design the controller in the system. If the controller is designed for the maximum process gain, then it will ensure the stability of the system for all the gain values [22], [23]. Therefore to design a robust controller for the proposed system, we derive the c^i corresponding to maximum process gain (c_{max}) and later use it in the section V-E to design the controller.

The maximum gain c_{max} occurs when $B_{ji}^{fes} = B_{min}^{fes}$ and $k_{ik_{ovd}} = \frac{1}{R_{min}}$. B_{min}^{fes} is the minimum value of the B_{ji}^{fes}, among all the possible values of i and j, *i.e.*, $B_{min}^{fes} = \min_{i,j} B_{ji}^{fes}$. Another factor $k_{ik_{ovd}}$ as discussed in section IV-B, attains its maximum value when reputation of user is minimum. R_{min} is the minimum value of the reputation at which a user is eligible for resource allocation in the network. Therefore, the maximum value of $k_{ik_{ovd}}$ such that a user can receive data is $\frac{1}{R_{min}}$. Apart from these variables, the total capacity currently allocated by user i, *i.e.*, $\sum_{j\in\mathcal{Z}_i} r_{ji}$ is equal to its previous shared capacity S_i^{old} plus the controller's current output (y). We put all the above parameters in (14) to calculate the maximum gain. It is given as

$$c_{max} = \frac{d\left[\left(\frac{R_i^{old} + \frac{(S_i^{old}+y)}{l_i \times B_{min}^{fes}}}{2}\right) \times \frac{1}{R_{min}} \times \sum_{k\in\mathcal{A}_i} 1\right]}{dy}, \quad (15)$$

where $\sum_{k\in\mathcal{A}_i} 1$, represents the cardinality of the set $\mathcal{A}_i$ and is equal to g_i, the number of the users from which requester i is currently requesting the resources. In a network, the average generated (g_i) and received (l_i) requests by any user i settles down to the same value after some time *i.e.* $\frac{g_i}{l_i} \to 1$. This happens because a network cannot store data packets. Therefore average upload and download should be equal to maintain the network balance. Besides reputation system forces users to download in proportion to their contribution or upload. Consequently, the requests generated for average upload and download by a user should be the same. Thus we substitute $\frac{g_i}{l_i} = 1$ in equation (15) and get

$$c_{max} = \frac{d\left[\frac{1}{2} \times \left(R_i^{old} + \frac{(S_i^{old}+y)}{l_i \times B_{min}^{fes}}\right) \times \frac{g_i}{R_{min}}\right]}{dy} = \frac{1}{2R_{min}B_{min}^{fes}}. \quad (16)$$

Apart from gain c_{max}, change in the upload capacity does not change the output immediately. The change in output is observed after some delay (say T). For example, in discrete times systems which operate in discrete time periods, the resources are awarded to a user on the basis of its reputation (or the upload capacity) in the last period. This delay in output results in the introduction of an additional term dead time (delay) in the transfer function $C(s)$. Thus,

$$C(s) = c_{max} \cdot e^{-sT} = \left(\frac{1}{2R_{min}B_{min}^{fes}}\right) e^{-sT}. \quad (17)$$

The overall transfer function of the control loop consisting of controller, actuator, plant, and monitor (refer Fig. 2b) is

$$T(s) = C(s)G(s) = c_{max} \cdot e^{-sT} K_p \left(1 + \frac{K_i}{s}\right). \quad (18)$$

For the ease of analysis, we have converted the above equation into the Fourier form as follows (Refer Appendix for details).

$$T(w) = c_{max} \cdot e^{-jwT} K_p \left(1 + \frac{K_i}{jw}\right). \quad (19)$$

This $T(w)$ is used in the next subsection to calculate the gain parameters K_p and K_i of the PI controller.

E. TUNING PI CONTROLLER

The proposed PI controller is tuned w.r.t. $C(s)$ on the basis of gain margin specification. The gain margin (G) [24] is the amount of increase or decrease in gain required to make the

[4]Process dynamics play a significant part during transient phase ($U\neq 1$) of the system. The transient state analysis is an interesting problem and will be taken up by the authors in future.

loop gain $T(s)$ equals to 1 at the phase crossover frequency (w_p), *i.e.*,

$$G = \frac{1}{|T(w_p)|} = \frac{1}{|c_{max}|\left|(e^{-jw_pT})(K_p)\left(1+\frac{K_i}{jw_p}\right)\right|}, \quad (20)$$

where w_p is defined as the frequency where phase angle $(\angle T(w_p))$ is $-\pi$. Thus,

$$\angle T(w_p) = -w_pT - tan^{-1}\left(\frac{K_i}{w_p}\right) = -\pi. \quad (21)$$

In the industrial PI controllers, it is common practice to tune the controller phase i.e $-tan^{-1}\left(\frac{K_i}{w_p}\right)$ to $-\frac{\pi}{6}$ [22], [25], thus

$$tan^{-1}\left(\frac{K_i}{w_p}\right) = \frac{\pi}{6}. \quad (22)$$

Using (22) in (21) we get the value of phase crossover frequency as $w_p = \frac{5\pi}{6T} = \frac{2.618}{T} rad/s$.

We substitute the value of w_p in (22) to calculate K_i as

$$K_i = \frac{1.512}{T}. \quad (23)$$

The above computed values of w_p and K_i are used in (20) for obtaining the expression for K_p as

$$K_p = \frac{0.866}{c_{max}G}. \quad (24)$$

The parameter G and T are set by the designer and c_{max} is calculated using (16). The equations (23) and (24) provide the generic value of the model parameters. The actual parameter values used during simulations are calculated in the next section. All the component values namely U_{ref}, $G(s)$ and $C(s)$ constituting the proposed control system in Fig. 2b, have also been evaluated. The subsequent section provides details about the practical realization and evaluation of the proposed system.

VI. PERFORMANCE EVALUATION

In the previous section, we have calculated gain parameters K_p and K_i. Due to ease of analysis, we have considered a continuous system for calculating these gain parameters. However in practical systems, the parameter (*i.e.*, capacity) used for performance analysis changes in discrete steps at discrete time instants. Therefore we have simulated a discrete time network for analyzing the performance of the proposed model. In this discrete model, we have used the values of gain parameters calculated for the continuous case. The simulation results presented in subsection VI-B show that the output obtained by using these gain parameters in the discrete time model closely follows continuous system response. Further, simulation results in subsections VI-C and VI-D illustrate that the discrete system using these values successfully operate at the optimal partitioning level. As there is no significant loss of data or introduction of spurious information at the output, so use of K_p and K_i values derived for a continuous case, in the discrete system are acceptable.

TABLE 1. Simulation parameters with their values.

Parameter	Description	Value
K_p	Proportional Gain	0.00577
K_i	Integral Gain	0.151149
C_i	The access link capacity at the user i	4 Mb/s
G	Gain Margin	3
U_{ref}	Reference value of Level of optimality	1
R_{in}	Reputation assigned to newcomers	0.07
R_{min}	Reputation threshold below which a user cannot receive any service	0.01
B^{fes}_{min}	The minimum of feasible bandwidth of all the links possible in network	2 Mb/s

We consider a discrete system which advances in periods of fixed time duration. Each period is assumed to last for 10 *sec* as in [6]. As changes initiated by controller appears at the output with delay of 1 period, so dead time T (refer (19)) is taken as 10 *sec* in the current simulation. At the start of a period using the partitioning mechanism, every user divides its total link capacity between upload and download. After that, requesters send the requests to the other users and if they get selected for the service allocation then their requests get fulfilled within the same period. Based on the service provided by the serving user, the requester calculates its reputation at the end of the period, while new entrants are provided with an initial reputation ($R_{in} = 0.07$) for their survival. Depending on the reputation value, a user will be awarded services in the next time period. However if a user's reputation falls below the threshold reputation ($R_{min} = 0.01$), then it is rendered ineligible for receiving services. After the reputation of all the users is updated, the next period will start. For the next period, the whole process will be repeated again.

To ascertain the performance of the capacity partitioning schemes, authors in [6], [10], simulated a network of 50 or 100 users. However, to demonstrate that the proposed system will work fine even for a larger number of users, we have simulated a network with 1000 users. To facilitate readability, various simulation parameters along with their values are listed in Table 1. As in [24], the gain margin of the system is set to $G = 3$, whereas the remaining parameter values are taken from the models used in [6], [10]. Directly using parameter values from the existing models allows us to compare our results with theirs. Finally we use these parameter values in (23) and (24) to calculate $K_p = 0.00577$ and $K_i = 0.151149$ respectively.

Using the above-mentioned specifications together with the mechanism for resource discovery and exchange discussed under section III, we have developed a customized simulator for evaluating the performance of different partitioning mechanisms in the distributed network. We now present implementation of the proposed capacity partitioning mechanism in the real world network.

A. IMPLEMENTATION OF THE CONTROL MODEL

The proposed partitioning mechanism based on a control theoretic approach is an algorithm, which we name as adaptive step size (ASZ). A user runs ASZ to dynamically adjusts its upload and download capacity, so that it operates around optimal point of partitioning.

If i has just joined the network, ASZ will allocates its link capacity equally between upload and download, *i.e.*, $S_i^0 = D_i^0 = \frac{C_i}{2}$. Here superscript 0 denotes the period of operation. Such division results in better reception of resources from the network (refer next subsection for details). Otherwise in any other period p, ASZ divides the C_i such that it minimizes the deviation of the of current partitioning level (U_i^p) from the optimal level (U_{ref}). The deviation for the period p is stored in a variable $E_i^p = U_{ref} - U_i^p$. E_i^p along with the summation of all the deviations that have occurred till period $p-1$, *i.e.*, TE_i^{p-1} is used to compute the output (y_i^p) of the PI controller. The output of the PI controller automatically corrects the system output such that it minimizes deviation E_i^p [21], *i.e.*, it makes the system partitioning level approach the optimal level. Due to the proportional action ($K_p \times E_i^p$) of the PI controller a user quickly achieves the optimal level of partitioning. When the system reaches optimal partitioning state, integral action ($K_i \times (E_i^p + TE_i^{p-1})$) maintains user's level of optimality around the optimal value. Initially when the deviation is more, y_i^p is large, but as the user reaches optimal state deviation $\rightarrow$ 0 and so does the y_i^p. In the current system, the controller achieves the auto-correction by modifying the upload capacity in terms of its output (y_i^p). The capacity allocated for the upload in the next iteration, *i.e.*, (S_i^{p+1}) gets modified by amount y_i^p. However, if the change in upload capacity reduces the download bandwidth (D_i^{p+1}) below a threshold value (Δ_{thr}), then the upload and the download capacities are set to values $C_i - \Delta_{thr}$ and Δ_{thr} respectively. This is done so that user is capable of receiving at least a minimum level of resources (Δ_{thr}) from the network. Finally, the new partition values, *i.e.*, S_i^{p+1} and D_i^{p+1} are updated by the user for the next period $p+1$. The above process is repeated as long as the user remains in the network.

Thus, ASZ changes the upload capacity in the step size (y_i^p) which is a function of deviation from the level of optimality from the reference point. Due to dynamically adjusting step size, the upload capacity and thereby the user's level of optimality stabilizes around the optimal point, unlike the existing mechanisms [6] where fixed step size causes shared capacity to oscillate around the optimal point. The ASZ algorithm is presented in Algorithm 1. In the subsequent sections, we study the performance of the ASZ algorithm in different operational scenarios.

B. STUDY OF CONTROL SYSTEM PERFORMANCE

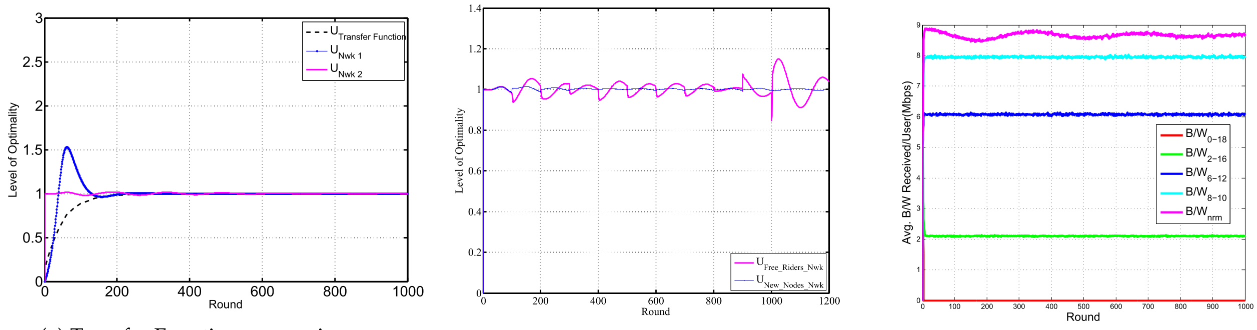
In this section, we compare the performance of the continuous time linear model of partitioning process with the actual discrete time distributed network, to verify the following: Accuracy of the transfer function, and appreciable change in output when parameter derived for the linear model are used in the actual distributed system.

Algorithm 1 ASZ: Bandwidth Allocation Algorithm

```
Initialization:
Initialize $K_p$, $K_i$ and $C_i$ from Table 1
Set $S_i^0 = \frac{C_i}{2}$, $\Delta_{thr} = \frac{C_i}{10}$, $E_i^0 = 0$, $TE_i^{-1} = 0$ and $U_{ref} = 1$
$p \leftarrow 0$
Shared Capacity Evaluation:
repeat
    Compute $U_i^p$ as in equation (7) and update
    $E_i^p \leftarrow U_{ref} - U_i^p$
    $y_i^p \leftarrow K_p \times E_i^p + K_i \times (E_i^p + TE_i^{p-1})$
    $TE_i^p \leftarrow TE_i^{p-1} + E_i^p$
    $S_i^{p+1} \leftarrow S_i^p + y_i^p$
    if $C_i - S_i^{p+1} \leq \Delta_{thr}$ then
        $S_i^{p+1} \leftarrow C_i - \Delta_{thr}$
    end if
    $D_i^{p+1} \leftarrow C_i - S_i^{p+1}$
    $p \leftarrow p+1$
    return $S_i^{p+1}$ and $D_i^{p+1}$
until User $i$ is in the network
```

Using transfer functions $G(s)$ and $C(s)$ derived in section V, we have simulated a linear model of capacity partitioning process in SIMULINK. Fig. 2b presents the block diagram model used in SIMULINK. The components values used in this model, *i.e.*, transfer functions $G(s)$ and $C(s)$ are evaluated from (8) and (17) respectively. The parameter values used in these equations are listed in Table 1. Apart from the linear model, we have also simulated two distributed network viz., Nwk 1 and Nwk 2, where ASZ is used to decide the partitioning of the link capacity. These distributed networks differ in the link capacity partitioning followed by the newcomer when it joins the network. At the outset in the Nwk 1, a new entrant allocates 0 and entire link capacity for the upload and download respectively, whereas in the Nwk 2, a new entrant allocates its half of the link capacity for both upload and download. After that, members of both the networks use ASZ to adjust their link capacity.

The level of optimality achieved by the linear model, Nwk 1, and Nwk 2 are represented by $U_{Transfer\ Function}$, $U_{Nwk\ 1}$, and $U_{Nwk\ 2}$ respectively in Fig. 2b. Initially, there is a slight mismatch in the level of optimality obtained by users in these three systems. This happens because users in the linear model and the Nwk 1 initially share nothing, while users in Nwk 2 share half of the link capacity for upload. Therefore members in the linear model and Nwk 1 initially receive the level of optimality equal to 0, whereas in Nwk 2 members initially receive the level of optimality close to the optimal level of partitioning (U_{ref}). It takes some time for users in the



(a) Transfer Function comparison with actual network

(b) System performance under changing Network Dynamics

(c) Comparability Analysis of ASZ

FIGURE 3. Performance evaluation of ASZ.

linear model and Nwk 1 to reach U_{ref}. This perturbation is more prominent in Nwk 1, where initially $U_{Nwk\ 1}$ is 0, then it overshoots U_{ref} and after some time return back to U_{ref}. Initially, as all the users in Nwk 1 share nothing, so they receive 0 resources resulting in $U_{Nwk\ 1} = 0$. The error input $(U_{ref} - U_{Nwk\ 1})$ to controller becomes very high. This causes PI output to reach very large value for the next period, which leads to users allocating very larger part of their link capacity for upload. After some time, resources to be downloaded are available in abundance across the network. So $U_{Nwk\ 1}$ overshoots U_{ref}. The PI controller again readjusts its output to modify the shared capacity so that $U_{Nwk\ 1}$ stabilizes around U_{ref}. The initial deviation in Nwk 1 is more than in the linear model because we had assumed a linear system while deriving controller (K_p and K_i) parameters, which is not the case with Nwk 1. In spite of these initial deviation, for most of a user's lifetime output of the three systems are in argument with each other. Hence these deviations can be tolerated. This confirms the accuracy of the transfer function. It also validates that the PI controller tuning parameters (K_p and K_i) derived earlier for the continuous case (linear model) can be successfully applied for the discrete time systems (Nwk 1 and Nwk 2). The linear model is just for evaluating K_p and K_i, while the practical systems are modeled using Nwk 1 and Nwk 2.

Among Nwk 1 and Nwk 2, users in Nwk 2 initially partition their link capacity equally between upload and download. So unlike Nwk 1, level of optimality of members of Nwk 2 reaches U_{ref} at the starting of the network. Thus as in Fig. 2b, linearity condition of a user which is always working around the optimal point is never violated in Nwk 2. Hence, when compared with members of Nwk 2, initially users of Nwk 1 are at a loss because they do not operate at the optimal partitioning point. Any user will prefer Nwk 2 over Nwk 1. Therefore we use Nwk 2 in the remaining portion of the paper for simulation and comparison purpose. This justifies the following assumptions- Linearization of the system around the optimal point for deriving transfer function (section V-D.2), and the initialization of the upload and download capacity of a new user as half of the link capacity (Algorithm 1).

C. ASZ'S ADAPTIVENESS ANALYSIS

In this section, we study the adaptiveness of ASZ to the changing network dynamics. Network's dynamic changes when either the new users entering or the existing users leaving the network or becoming free-riders–this changes resources available for download across the network–which may change the optimal point for a user. Through simulation, we determine ASZ ability in handling the change in the optimal point.

We first determine how a user's level of optimality changes in the presence of ASZ changes, when the new entrants arrive in the network. A network with 100 users initially is simulated to study this change. After the end of the periods which are multiples of 100, 100 new users are assumed to arrive in the network. It implies that when 100th and 200th period ends, the total number of users in the network become 200 and 300 respectively. This process continues till 900th period, where total count of members in network reaches 1000. All the users in the network use ASZ to partition their link capacity. With *ASZ*, a new user i start uploading $(\frac{C_i}{2})$ as soon they enter the network. This ensures that the net resources demanded is equal to resources available across the network. As equilibrium between demand and resources available is maintained, so there should not be appreciable change in the output of this system ($U_{New\_Nodes\_Nwk}$). Simulation result in Fig. 3b also substantiates this claim. $U_{New\_Nodes\_Nwk}$ almost remain close to the optimal level (U_{ref}) as the new users join the network. Hence, the new users entering the system do not disturb the optimal point of operation of the other users.

We now ascertain the robustness of ASZ in handling free-riders and the users leaving the network. A user becoming free-rider is more harmful than the departing users. Departing users will neither contribute nor consume resources, whereas free-riders in spite of no contribution, will try to consume resources. So users becoming free-rider will create a greater demand and resource availability imbalance in comparison to the users leaving the network. Thus, a partitioning mechanism capable of maintaining an optimal level of partitioning in presence of free-riders can successfully handle the departure of users. So we check ASZ's performance only for free-riding users.

In the simulation model, the percentage of free-riders is gradually increased from 0% to 99.99% of the total users. Network starts with 1000 user with no free-riders. At the end of periods, which are multiple of 100, free-riders percentage is increased by 10% of the initial number of users. Thus, after 100^{th} period 10% contributing users become free-riders, it becomes 20% after 200^{th} period and so on. At the end of 1000 periods, 90% of the users free-ride. Finally to model worst- case scenario, after 1050 periods, 99.99% of the total users in the network are considered to be free-riding.

Fig. 3b, shows that the contributing users (non free-riders) do maintain their level of optimality ($U_{Free\_Riders\_Nwk}$) around optimal value (U_{ref}), even when some of the users are turning free-riders. $U_{Free\_Riders\_Nwk}$ do get perturbed and stabilizes again when a spurt of conversion to free-riders happen. During such a spurt, the supply of resources sees a sudden reduction while the demand remains the same, causing a decrease in $U_{Free\_Riders\_Nwk}$. As the reputation system reduces the reputation of the users converted to free-riders, after a while demand-supply balance is restored and $U_{Free\_Riders\_Nwk}$ achieves U_{ref}. At the same time, genuine users also try to increase the supply of resources to compensate for the deficit of uploaded resources caused by free-riders. This leads to a brief oscillatory behavior in $U_{Free\_Riders\_Nwk}$ whose magnitude increases with increase in the fraction of free-riders. However, oscillation settles down after some time, and $U_{Free\_Riders\_Nwk}$ stabilizes around U_{ref}. Thus, simulation demonstrated that ASZ is adaptive to dynamics changes in the network.

D. COMPATIBILITY ANALYSIS OF CONTROL SYSTEM

In a distributed network, every user is free to use its own partitioning mechanism. Each partitioning mechanism should be compatible with the other mechanisms or strategies being used to achieve optimal partition. It is important to analyze the effectiveness of ASZ when it is deployed together with other partition strategies. We assume that ASZ is implemented at each user independent of the partitioning strategies employed by the other users. It will optimize on the basis of resources received from the other users in the network. Therefore, analyzing performance of ASZ w.r.t. efficient partitioning strategies[5] [6], [9] is almost similar to analysis, where all the users are employing ASZ. In such a scenario, all the users will be able to easily achieve and subsequently maintain their level of optimality close to the optimal point. The real performance issue lies in more adverse scenario where net resources available across network is less than resources demanded. Therefore we consider a scenario with scarcity of resources, *i.e.*, other users are either free-riders or employ inefficient strategies, which always allocate greater part of its link capacity for download, independent of the resources received by the user. The simulation model to analyze the performance of ASZ is described below.

Every user is connected to the network with a single capacity link of 18 Mb/s as in [6]. In the network, 20% of users are normal users, who partition their capacity using ASZ. Their corresponding received bandwidth is denoted by BW_{nrm}. The users employing inefficient partitioning strategies are modeled as follows. 20% of users completely free-ride, *i.e.*, they use their entire link capacity of 18 Mb/s for download, other 20% users will contribute at most 2 Mb/s of their capacity for uploading, while another 20% will provide maximum 6 Mb/s for upload and the remaining 20% users, will provide maximum 8 Mb/s of their link capacity for upload. The corresponding bandwidth received by these set of users is denoted by BW_{0-18}, BW_{2-16}, BW_{6-12}, and BW_{8-10} respectively. All these users are designated as free-riders, as they request more and share less.

From Fig. 3c, it is evident that members who use ASZ receive maximum bandwidth from the network. The amount of bandwidth received by them does get affected, as there is a scarcity of resources across the network because of the high level of free-riding (80% of the users free-ride in some form). Due to the resource scarcity, initially allocating half of their link capacity for upload is not enough for users to achieve optimal partitioning. Hence, an initial oscillatory behavior is observed in BW_{nrm} for users employing ASZ, which eventually settles down. As other users are not using ASZ for capacity partitioning, so they do not exhibit oscillatory behavior. However, these users are at a loss as the bandwidth received by them decreases with the decrease in the allocation for the upload capacity—due to a decline in their contribution level—although they have more download bandwidth available at their disposal. Thus, there is no advantage for users in allocating more towards download. Since users employing ASZ receive highest resources, so all the users in the network will gradually start using it.

Clearly, ASZ is compatible with other partitioning strategies, and users will eventually shift towards using it.

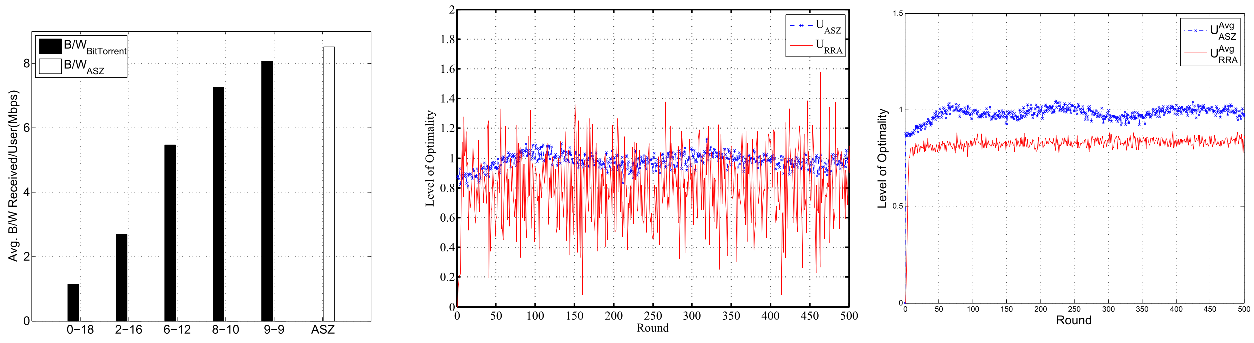
VII. COMPARISON WITH EXISTING SCHEMES

Until now, we evaluated ASZ in the general setup for distributed networks and explored its effectiveness in achieving optimal capacity partitioning. In this section, the existing schemes: BitTorrent [10] and Reputation-Based Resource Allocation (RRA) Policy [6] are compared with ASZ.

A. COMPARISON WITH BITTORRENT

BitTorrent is a popular file sharing protocol which is also based on the concept of cooperative computing. It lacks capacity partitioning algorithm [6] to partition link capacity between upload and download in single capacity links. We study the change in received resources when ASZ is integrated with BitTorrent.

[5]The strategies in which average bandwidth allocated for upload and download are close to optimal partition are referred as efficient partitioning strategy



(a) BitTorrent Comparison (b) RRA Comparison at single user (c) RRA Comparison averaged over network

FIGURE 4. Comparison of ASZ with Existing schemes.

1) SIMULATION SETUP

We simulate an elementary version of BitTorrent based on Azures, a popular BT Client [6], [10]. Since a typical swarm (Set of users actively uploading and downloading data) in BitTorrent consists of around 50 users, we have also simulated the same sized network for comparison of ASZ with the BitTorrent. All the users in this network are connected by a single capacity link of 18 Mb/s. As this paper focuses on evaluating the performance in terms of efficient link capacity partitioning, so we overlook the chunk selection algorithm and assume that a user is always interested in chunks of files available with its neighbor. The users request for chunks at the starting of a period. After receiving requests, a user x implements choking algorithm [10] for its resource distribution. Top four requesters who have provided the highest download rate to x are selected for service. Apart from this, after every third period, x randomly selects a requester for service regardless of its download performance. This allows new entrants to obtain initial chunks, and user x to explore other users in the network. Once x finalizes the requesters, then it allocates its upload capacity equally among them.

2) BITTORRENT LIMITATIONS AND RESULT COMPARISON

We now illustrate how the absence of capacity partitioning algorithm in BitTorrent leads to a reduction in resources received by users. A network is simulated, where users are equally divided into five groups. Members in group $0-18$, $2-16$, $6-12$, $8-10$ and $9-9$ allocate 0, 2, 6, 8, and 9 Mb/s respectively for upload. It is evident from Fig. 4a that users receive very less amount then what they upload. Average resources received by users across the network is equal to 4.9 Mb/s.

To enhance the received resources, ASZ is integrated with BitTorrent. At the start of a period, every BitTorrent user uses ASZ for deciding the division of its link capacity. The ASZ strategically divide the link capacity between upload and download so that a user receives maximum resources from the network. This is also validated by the simulation results in Fig. 4a, where the average bandwidth received increases from 4.9 to 8.5 Mb/s.

B. ASZ COMPARISON WITH REPUTATION-BASED RESOURCE ALLOCATION (RRA) POLICY

RRA [6] is a capacity partitioning algorithm for distributed systems which dynamically divides link capacity between upload and download so that a user can receive maximum resources from the network. We know compare the performance of RRA with ASZ. The performance comparison is in terms of the resources received by the users.

1) SIMULATION SETUP

The simulation model used for comparing RRA with ASZ is based on the model in [6]. A 100 member distributed network is simulated. Let C_i, S_i^p and D_i^p represent the total, upload and download capacity of the user i during time period p. Users initially upload nothing; during subsequent periods they change their upload capacity in fixed step size $\Delta = \frac{C_i}{10}$. The increase in upload capacity increases user's chances of getting more resources. When the resources allocated (T_i^p) to user i is less than its download capacity (D_i^p), it increases its upload capacity (S_i^{p+1}) by Δ amount in the next period. Otherwise if $T_i^p \geq D_i^p$, then RRA further try to maximize resources received by increasing its D_i^{p+1} by Δ at the expense of decreased S_i^{p+1}. To model RRA algorithm as a control system, "*If*" conditions in original RRA algorithm [6] are re-written in terms of level of optimality (U_i) for a user i. If $T_i^p < D_i^p$, level of optimality $(U_i^p = T_i^p/D_i^p)$ is less than 1; so for $U_i^p < 1$, S_i^{p+1} is incremented by Δ. Similarly if $T_i^p > D_i^p$, then $U_i^p \geq 1$, resulting in decrement in S_i^{p+1} by Δ. The RRA in its new form is represented by Algorithm 2.

2) RRA MODELING AS CONTROL SYSTEM

The capacity partitioning process using RRA as control system is shown in Fig. 2c. The transfer function $(C(s))$ derived in the section V-D.2 is reused to represent the actuator and the plant. As the controller's output changes the upload capacity and thereby the capacity partitioning of a user, so the controller is specific to the partitioning algorithm being used. The input of the RRA controller is the error signal: $U_{ref} - U_i$. For $U_i < 1$ error signal is positive, and for $U_i > 1$ it is negative. From algorithm 2, the output of RRA controller

Algorithm 2 RRA Bandwidth Allocation algorithm

```
Initialization:
Initialize $C_i$ from Table 1
Set $S_i^0 = 0$, $\Delta = \frac{C_i}{10}$ and $U_{ref} = 1$
$p \leftarrow 0$
Shared Capacity Evaluation:
repeat
    Compute $U_i^p$ as in equation (7) and update
    if $(U_i^p < 1)$ AND $(S_i^p < C_i - \Delta)$ then
        $S_i^{p+1} \leftarrow S_i^p + \Delta$
    else if $(U_i^p \geq 1)$ AND $(S_i^p > \Delta)$ then
        $S_i^{p+1} \leftarrow S_i^p - \Delta$
    end if
    $D_i^{p+1} \leftarrow C_i - S_i^{p+1}$
    $p \leftarrow p + 1$
    return $S_i^{p+1}$ and $D_i^{p+1}$
until Node $i$ is in the network
```

for positive and negative error is $+\Delta$ and $-\Delta$ respectively. Hence $G_{RRA}(s)$, the transfer function of RRA controller is ($\Delta \times$signum function) as shown in Fig. 2c. Laplace transform of signum function is given by $\frac{2}{s}$ [26], so

$$G_{RRA}(s) = \frac{2\Delta}{s}. \tag{25}$$

Since the derived controller is not tuned keeping system's stability under consideration (Gain and Phase margin are not used to derive the controller), so the simulation of the linear model (i.e., $T(s) = G_{RRA}(s) \times C(s)$) of the system using SIMULINK, leads to unbounded output U_i, oscillating between $(-\infty, \infty)$. In the actual network simulation, as the link capacities are finite, so U_i does not reach ∞, but it saturates and oscillates around the optimal point with a finite amplitude. This reduces resources received by a user when it implements RRA instead of ASZ. Reduction in the performance will be explained in detail in the next subsection.

3) RRA RESULT COMPARISON AND LIMITATION

Let U_{RRA} and U_{ASZ} denote the level of optimality received by users when they use RRA and ASZ respectively. Fig. 4b and 4c presents the results comparing RRA with ASZ. While Fig. 4b shows level of optimality (U) observed at a single user, Fig. 4c plots level of optimality $\left(U^{Avg}\right)$ which is averaged out across all the users in the network.

One user is randomly selected from the network and U_{RRA} and U_{ASZ} is plotted in Fig. 4b. Simulation results show that U_{RRA} never becomes equal to U_{ref}, but oscillates around it. This behavior results from the fact that RRA uses a fixed step size Δ for modification in the capacity partitioning. Thus U_{RRA} never settles down and continues oscillating around U_{ref} with amplitude proportional to Δ. However, ASZ employs the integral action [21] of the PI controller to adjusts the step size and reduce the steady state error.[6]

[6] Steady state error is equal to the difference between U_{ref} and level of optimality received, when the system is operating near U_{ref}.

Due to adaptive step size, a user maintains its U_{ASZ} close to U_{ref}. Satsiou and Tassiulas [6] who proposed RRA studied average bandwidth allocation for the overall network. They summed the bandwidth received by all the users and then found the average. Since the results provided only the mean and not the higher order moments like variance, they were unable to show the unsettled behavior in the network. It physically implies that the average across the overall network gives a false impression that bandwidth has stabilized. It happens because users whose capacity increases get compensated by the users whose capacity decreases.

The unsettled behavior in the received bandwidth for a single user is also reflected in the simulation results averaged out across the network. In Fig. 4c, the average level of optimality (U_{RRA}^{Avg}) achieved using RRA algorithm stabilizes at a point lower than U_{ref}. U_{RRA}^{Avg} is unable to achieve U_{ref} because of the greater wastage of resources in the network implementing RRA. Bandwidth wastage occurs because users do not operate close to U_{ref}. When $U_{RRA} > U_{ref}$ for a user, it is allocated more resources than its download capacity. This user cannot utilize the resources greater than its download capacity. At the same time, a needy user with $U_{RRA} < U_{ref}$ receives less than its download capacity, and will be deprived of the resource. ASZ minimizes bandwidth wastage by using adaptive step sizes, and U_{ASZ}^{Avg} remains around U_{ref} for most of network lifetime (refer Fig. 4c). Therefore in terms of the received resources, the performance of ASZ is better than RRA.

4) COMPLEXITY COMPARISON

The PI parameters calculation is a one time process: Its time complexity is $\mathcal{O}(1)$. The number of arithmetic operations and storage variables required in ASZ is more than in RRA, but their asymptotic bound is same: The time and space complexity for both the algorithms is $\mathcal{O}(n)$. Hence, ASZ algorithm gives better performance than RRA, without any significant increase in the system's complexity.

VIII. CONCLUSIONS

We have presented a complete framework to analyze and solve the problem of optimal partitioning of link capacity in single capacity links. A metric "level of optimality (U)" is introduced to determine how close to the optimal partitioning, a given partition mechanism operates. $U = 1$ corresponds to the optimal partitioning level (U_{ref}). To achieve U_{ref}, we have modeled capacity partitioning as a feedback control problem and proposed an algorithm (ASZ). This algorithm is adaptive to network changes, and users can maintain their U close to U_{ref} in the presence of the arrival and the departure of other users from the network. ASZ is also compatible with existing partitioning schemes. When a portion of users in the network employ some other partitioning technique, it is still able to optimally divide the link capacity for a user.

We have also evaluated ASZ performance with other distributed protocols, viz., BitTorrent and Reputation-Based Resource Allocation(RRA). BitTorrent lacks any mechanism to partition link capacity optimally. ASZ was integrated with

BitTorrent and users were able to receive more resources by operating at the optimal partitioning level. The ASZ was also compared with RRA. Since ASZ operates closer to the optimal partitioning point compared to RRA, resources received by a user employing ASZ were more.

APPENDIX

LAPLACE TO FOURIER CONVERSION OF TRANSFER FUNCTION

The Laplace transform of any function $f(t)$ is $\int_0^t f(t)e^{-st}dt = \int_0^t f(t)e^{-(\sigma+jw)t}dt$, where $s = \sigma + jw$. The real part σ contributes to the term $e^{-\sigma t}$. This term decays to zero during steady state ($t \to \infty$) and only the jw part which gives the sinusoidal steady state response (*i.e.*, $e^{-jwt} = cos(wt) - jsin(wt)$) remains. Life time of a user in the network is very large when compared with its initial bootstrapping period. Hence it can be assumed, that a user mostly operates in the steady state. Therefore σ can be neglected, and $s = jw$ can be substituted in the overall transfer function of the control loop.

REFERENCES

[1] K. Shin, C. Joe-Wong, S. Ha, Y. Yi, I. Rhee, and D. S. Reeves, "T-Chain: A general incentive scheme for cooperative computing," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2122–2137, Aug. 2017.

[2] J. Berry, M. Collins, A. Kearns, C. A. Phillips, J. Saia, and R. Smith, "Cooperative computing for autonomous data centers," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, May 2015, pp. 38–47.

[3] E. Cerqueira, E. Lee, J.-T. Weng, J.-H. Lim, J. Joy, and M. Gerla, "Recent advances and challenges in human-centric multimedia mobile cloud computing," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2014, pp. 242–246.

[4] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.

[5] M. Feldman and J. Chuang, "Overcoming free-riding behavior in peer-to-peer systems," *ACM SIGECOM Exchanges*, vol. 5, no. 4, pp. 41–50, Jul. 2005.

[6] A. Satsiou and L. Tassiulas, "Reputation-based resource allocation in p2p systems of rational users," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 4, pp. 466–479, Apr. 2010.

[7] D. G. Jeong and W. S. Jeon, "CDMA/TDD system for wireless multimedia services with traffic unbalance between uplink and downlink," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 5, pp. 939–946, May 1999.

[8] C.-H. Chiang, W. Liao, and T. Liu, "Adaptive downlink/uplink bandwidth allocation in IEEE 802.16 (WiMAX) wireless networks: A cross-layer approach," in *Proc. IEEE Global Telecommun. Conf. (IEEE GLOBECOM)*, Nov. 2007, pp. 4775–4779.

[9] M. Meo and F. Milan, "A rational model for service rate allocation in peer-to-peer networks," in *Proc. IEEE INFOCOM.*, Apr. 2006, pp. 1–5.

[10] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. Workshop Econ. Peer-Peer Syst.*, Jun. 2003, pp. 68–72.

[11] D. Banerjee, S. Saha, S. Sen, and P. Dasgupta, "Reciprocal resource sharing in P2P environments," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst.*, 2005, pp. 853–859.

[12] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in P2P networks," in *Proc. Int. Conf. World Wide Web*, 2003, pp. 640–651.

[13] G. Iosifidis and I. Koutsopoulos, "Reputation-assisted utility maximization algorithmsfor peer-to-peer networks," in *Proc. Int. Workshop Qual. Service*, Jun. 2008, pp. 20–29.

[14] N. Singha and Y. N. Singh, "Optimal capacity partitioning in homogeneous P2P network," *IEEE Commun. Lett.*, vol. 22, no. 7, pp. 1354–1357, Jul. 2018.

[15] H. Zhang, Y. Wen, H. Xie, and N. Yu, *Distributed Hash Table: Theory, Platforms and Applications*. New York, NY, USA: Springer, 2013.

[16] W. Ai, L. Xinsong, and L. Kejian, "Efficient flooding in peer-to-peer networks," in *Proc. 7th Int. Conf. Comput.-Aided Ind. Design Conceptual Design*, Nov. 2006, pp. 1–6.

[17] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer, "Free riding in BitTorrent is cheap," in *Proc. 5th Workshop Hot Topics Netw.*, Nov. 2006, pp. 85–90.

[18] R. T. B. Ma, S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau, "Incentive and service differentiation in P2P networks: A game theoretic approach," *IEEE ACM Trans. Netw.*, vol. 14, no. 5, pp. 978–991, Oct. 2006.

[19] Newcastle University. *Exponential Moving Average*. Accessed: Oct. 18, 2018. [Online]. Available: http://www.lorien.ncl.ac.uk/ming/filter/filewma.htm

[20] R. Gupta and Y. N. Singh, "Reputation aggregation in peer-to-peer network using differential gossip algorithm," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 10, pp. 2812–2823, Oct. 2015.

[21] K. Ogata, *Modern Control Engineering*, 5th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2011.

[22] T. F. Abdelzaher, K. J. Shin, and N. Bhatti, "Performance guarantees for Web server end-systems: A control-theoretical approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 1, pp. 80–96, Jan. 2002.

[23] R. Berber and C. Kravaris, *Nonlinear Model Based Process Control* (Applied Sciences Series). Norwell, MA, USA: Kluwer, 1998.

[24] W. K. Ho, C. C. Hang, and L. S. Cao, "Tuning of PID controllers based on gain and phase margin specifications," *Automatica*, vol. 31, pp. 497–502, Mar. 1995.

[25] F. G. Shinskey, *Process Control Systems: Application, Design, and Tuning*. New York, NY, USA: McGraw-Hill, 1996.

[26] A. V. Oppenheim, A. S. Willsky, and S. Hamid, *Signals and Systems*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

NITIN SINGHA received the Ph.D. degree from IIT Kanpur, Kanpur, India. He is currently an Assistant Professor and the Head of the Department of Electronics and Communication Engineering, Indian Institute of Information Technology, Kurnool, India. His areas of specialization are peer-to-peer networks, game theory, and block chain.

YATINDRA NATH SINGH received the Ph.D. degree from IIT Delhi, in 1997, with a focus on optical amplifier placement problem in all-optical broadcast networks. In 1997, he joined the EE Department, IIT Kanpur, where he was the Head of the Computer Centre and is currently a Professor of electrical engineering and the Dean of infrastructure and planning. He has supervised 14 Ph.D. and more than 97 M.Tech. theses so far. He has filed three patents for switch architectures and has published many journal and conference research publications. He has also written lecture notes on digital switching, which are distributed as open access content through content repository of IIT Kanpur. He has also been involved in open-source software development. He has started Brihaspati initiative, an open-source learning management system, BrihaspatiSync, a live lecture delivery system over Internet, and BGAS, general accounting systems for academic institutes. His research interests include telecommunications' networks, especially optical networks, switching systems, mobile communications, and distributed software system design. He is a Fellow of IETE, a Senior Member of the IEEE and ICEITE, and a member of ISOC. He received the AICTE Young Teacher Award, in 2003.

RUCHIR GUPTA received the Ph.D. degree from IIT Kanpur, Kanpur, India. He was an Assistant Professor with the Department of Computer Science and Engineering, Indian Institute of Information Technology, Jabalpur, India. He is currently an Assistant Professor with the Department of Computer Science and Engineering, IIT (BHU) Varanasi, India. His areas of specialization are peer-to-peer networks, social networks, game theory, NLP, and machine learning.