Computing and Informatics, Vol. 34, 2015, 402–424

IMPROVING REAL-TIME DATA DISSEMINATION PERFORMANCE BY MULTI PATH DATA SCHEDULING IN DATA GRIDS

Mustafa Müjdat Atanak, Atakan Doğan

Department of Electrical and Electronics Engineering
Anadolu University
26470 Eskişehir, Turkey
e-mail: {mmatanak, atdogan}@anadolu.edu.tr

Abstract. The performance of data grids for data intensive, real-time applications is highly dependent on the data dissemination algorithm employed in the system. Motivated by this fact, this study first formally defines the real-time splittable data dissemination problem (RTS/DDP) where data transfer requests can be routed over multiple paths to maximize the number of data transfers to be completed before their deadlines. Since RTS/DDP is proved to be NP-hard, four different heuristic algorithms, namely kSP/ESMP, kSP/BSMP, kDP/ESMP, and kDP/BSMP are proposed. The performance of these heuristic algorithms is analyzed through an extensive set of data grid system simulation scenarios. The simulation results reveal that a performance increase up to 8 % as compared to a very competitive single path data dissemination algorithm is possible.

Keywords: Data scheduling, data grids, real-time systems, simulation, performance evaluation

Mathematics Subject Classification 2010: 68M14, 68M20, 68U20

1 INTRODUCTION

Data grids are envisaged to run real-time, data-intensive applications emerging from many disciplines of science and engineering such as geographic information science, high-energy physics, and remote instrumentation [1, 2, 3]. These applications are

typically composed of a number of jobs, each of which is associated with a real-time requirement as well as a demand for access to terabytes/petabytes of data [4]. These data are usually read-only and produced by sensors of particle accelerators, globally positioned seismic sensors, or satellite images [4, 5]. For example, Compact Muon Solenoid (CMS) detector located at LHC (Large Hadron Collider) produces about 1 petabyte of read-only data every year. These data are stored at CERN and made available to researchers across the globe by means of the WLCG data grid system [5].

The data dissemination problem encountered in data grid systems such as WLCG deals with finding paths by which data will be transferred and with calculating bandwidths to be used for these transfers. Moving large-scale data on precomputed paths while providing bandwidth guarantees (quality of service – QoS – guarantees in general) has been the subject of many studies in the grid community as well as in the network community. In order to support end-to-end guaranteed service in the internet, for example, the IETF has defined integrated services (IntServ) architecture [6]. Later in [7], network element (router) behavior required to deliver a guaranteed delay and bandwidth in the Internet were described. Furthermore, Resource Reservation Protocol (RSVP) [8] complements IntServ by enabling the resource reservations on the routers along a path. In addition to these efforts, there are several research and education networks, e.g. Internet2 [9], ESnet [10], UltraScience Net [11], which have been enabled to provide on-demand or in-advance bandwidth guarantees [12].

While several different issues have been addressed in providing dedicated channels for data flows over multiple administrative domains in the literature, this study is mainly focused on the development of multi path data dissemination algorithms. Specifically, real-time splittable data dissemination problem (RTS/DDP), which formulates the problem of multi path data dissemination, is formally defined first. Four heuristic algorithms, namely kSP/ESMP, kSP/BSMP, kDP/ESMP and kDP/BSMP, are proposed after RTS/DDP is proved to be NP-hard. These algorithms try to schedule data transfer requests with real-time requirements through multiple paths in a data grid system with the objective of maximizing the satisfiability. The proposed algorithms are compared against very competitive single path and multi path algorithms in order to prove that they are beneficial to data grid systems in scheduling data transfer requests.

The rest of the paper is organized as follows: Section 2 surveys the related work. Section 3 formulates real-time splittable data dissemination problem. The solution of this problem is achieved in two phases. Section 4 presents the algorithms developed for the first phase, real-time k-path selection (RTPS-k). After finding a set of feasible paths for every request in the first phase, Section 5 introduces the algorithms proposed for the second phase, real-time flow sharing (RTFS), in order to yield the final data dissemination schedule. Section 6 shows a detailed set of simulation results related to four different multi path data dissemination algorithms. Finally, Section 7 concludes the study.

2 RELATED WORK

The data scheduling algorithms proposed in the literature can be grouped into two main categories, namely *single path* and *multi path data dissemination algorithm*. Furthermore, a single and multi path algorithm can schedule the data transfer requests in *on-demand*, *in-advance* or *batch* fashion.

In single path data dissemination, any data transfer is always routed using a single path between a source and destination pair while meeting some specific constraints. On-demand single path data dissemination algorithms (unicast QoS-based routing algorithms) have been extensively studied in the literature [13, 14, 15, 16, 17]. According to the characterization table in [17], QoS-based routing problems are classified into three classes, which are multiple constrained path, multiple constrained optimal path, and restricted shortest path. For each of these problems, a set of algorithms from the literature is reported in the table, where their common metrics are delay, jitter, bandwidth, packet loss, number of hops, link cost, etc. In addition, these QoS-based routing algorithms are usually extensions or modifications of well-known shortest path algorithms, e.g., Dijkstra, Bellman-Ford algorithms, so as to consider multiple QoS constraints.

With the advance of e-science applications [18] and next generation networks with bandwidth guarantees [12], *in-advance* and *batch single path data dissemination algorithms* have been proposed. In [19], the impact of a variety of advance reservation models on the in-advance single path data dissemination is investigated in detail. For the basic reservation model of [19], under which a path with minimum bandwidth value between a pair of source and destination nodes is searched for, the connection feasibility, maximum duration, and soonest completion problems are formulated, and they are all shown to be solvable in polynomial time. Later, the basic reservation model is extended to the advance cumulative reservation problem, where a path can transmit at any bandwidth value available through the network and adapt to the changes in available bandwidth. In [19], the advance cumulative reservation problem is shown to be NP-hard, and hence an approximation algorithm is proposed. In addition, similar algorithms to those in [19] for the basic reservation model are proposed in [20, 21]. In [21], nine different path computation algorithms are compared for the fixed slot in-advance scheduling problem, among which the *minimum hop feasible path algorithm* is shown to be the best in maximizing the network utilization. In [22], a few advance cumulative reservation problems, e.g., variable path with variable bandwidth, are formulated, and the algorithms for their solutions are provided.

Different from the on-demand and in-advance single path data dissemination algorithms in which data transfer requests are scheduled one after another, batch single path data dissemination algorithms schedule multiple data transfer requests concurrently in order to maximize the satisfiability, which is the ratio of the number of accepted requests to the total number of requests [23, 24, 25].

In the literature, there are other studies, e.g., [26, 27, 28, 29, 30], related to the data transfer scheduling using a single path in grid systems. In [26], a time-slot-based

approach for scheduling the elastic and streaming requests in Lambda Grids is described. In [27], the bulk data transfer scheduling (BDTS) problem, which searches for an optimal bandwidth allocation profile for each data request with deadline to minimize the overall network congestion, is defined. BDTS problem is proved to be solvable in polynomial time as a maximum concurrent flow problem. In [28], a general framework based on TCP is proposed to implement a bulk data transfer service in a grid network environment in which the bandwidth allocation profiles for data transfers are obtained as described in [27]. In [29], a novel queueing system is proposed to model the elastic data transfers with deadline in grid computing applications over a shared path. This model can be used to calculate the blocking probability of elastic data transfer requests, the necessary capacity of a bottleneck link, and the number of classes that can be supported in order to have the blocking probability of requests below a certain threshold. In [30], explicit admission control and high-speed transport protocols are combined to enable an opportunistic sharing of the network capacity by data transfer requests having heterogeneous bandwidth and delay requirements so that a high acceptance probability of such requests is achieved while maintaining efficient network-resource utilization.

Multi path data dissemination, which is a further generalization of the single path data dissemination, uses multiple paths simultaneously between source-destination pairs to deliver data at the related destinations while satisfying few specific constraints. Multi path data dissemination (routing) has been already supported by interior gateway protocols such as open shortest path first (OSPF) [31] and intermediate system to intermediate system (IS-IS) [32]. Apart from these connectionless protocols, multi-protocol label switching (MPLS) networks offer a connection-oriented approach [33]. That is, a label switched path is first set up between an ingress router and egress router, and then the ingress router encapsulates IP packets with labels that facilitate forwarding IP packets along the label switched path. In [34], a survey of deployed (OSPF, IS-IS, and MPLS) and incrementally deployable techniques that can provide internet-wide multi path routing is given. In [35], the maximum hop-count constrained traffic bifurcation problem is formulated for MPLS networks, which consists of finding hop-count constrained multiple paths between an ingress and an egress node pair with the objective of minimizing the maximum of link utilization, and a heuristic algorithm is proposed to achieve near-optimal link utilization values. In [36] and [37], the problem of multi path routing with bandwidth and delay constraints for a source-destination pair is studied, which requires finding one or more paths that meet the bandwidth constraint and minimize the delay of the longest path. [36] proposes a multi path heuristic algorithm that delivers video from a server to its client using a set of shortest paths in the max-flow subgraph of the related network, while [37] introduces a fully polynomial time approximation algorithm that performs better than the heuristic algorithm of [36]. In [38], k-splittable problem with end-to-end delay constraints (k-DCRP) for a source-destination pair is addressed, and a branch-and-price strategy with prohibitively long running times for large-scale networks is developed. In [39], restricted multipath and k-path routing problems are introduced as a network flow problem

with the objective of minimizing link congestion, and optimal and approximation algorithms are presented. In [40], the maximum concurrent file transfer problem with start and end times is formulated as the maximum concurrent flow problem whose objective is maximizing the throughput. The edge-path formulation of [40] results in a performance close to the optimal throughput with a reasonable time complexity while it uses a small number of pre-defined paths for each data transfer. In [41], several multipath reservation algorithms for in advance scheduling of single and multiple file transfers in connection-oriented optical networks are developed to produce schedules with minimum finish time. In addition to these studies, in the literature, there are other studies that address the different aspects of multi path routing, e.g., [42, 43, 44].

3 PROBLEM FORMULATION

A data grid system is modeled by an undirected graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ defines machines that include computing elements to run applications, storage elements to store application data, and network routers (or switches); $E = \{e_1, \ldots, e_m\}$ denotes links that connect any two machines, and $e_i \in E$ is associated with a bandwidth value $c_i > 0$ and a delay value $d_i \geq 0$. Network routers together with links form an interconnection network among machines.

$R = \{r_1, \ldots, r_\lambda\}$ denotes a set of real-time data transfer requests. Each request $r_i \in R$ is modeled by a quadruple $\langle s_i, t_i, f_i, \delta_i \rangle$ in which s_i is the source machine, t_i is the destination machine, f_i is the requested data item, and $\delta_i > 0$ is the deadline of request r_i.

$P_i = \{p_1, \ldots, p_{l_i}\}$ defines a set of paths for request $r_i \in R$, where $p_j \in P_i$ is a simple path which connects machines s_i and t_i; $l_i > 0$ is the number of such paths.

Definition 1. The bandwidth demand of request $r_i \in R$ on path $p_j \in P_i$, which is denoted by π_{ij}, is the minimum bandwidth value that guarantees the timely delivery of r_i at its destination.

$$\pi_{ij} = \frac{|f_i|}{\delta_i - \sum_{e_l \in p_j} d_l} \tag{1}$$

where $|f_i|$ denotes the data item size.

Definition 2. A path $p_j \in P_i$ is feasible for $r_i \in R$ if and only if $\beta_{ij} \geq \pi_{ij}$, where β_{ij} is the bottleneck bandwidth of path p_j and equal to the minimum of available bandwidth values of all the links that constitute the path.

Definition 3. A request $r_i \in R$ is satisfied if there exists at least one feasible path in P_i.

Definition 4. The satisfiability of a set of real-time data transfer requests is the number of satisfied requests in R by means of a scheduling algorithm.

In data grid systems, there is a *centralized data dissemination scheduler* in charge of making all real-time data scheduling decisions for all data transfer requests in R submitted by the running applications, e.g., [23, 24, 45]. Furthermore, the scheduler is capable of issuing reservation requests to the related components of the system in order for the data transfers to take place as scheduled, which is known as *advance reservation*. With respect to the scheduler and advance reservation models adopted by the system, when a data item f_i with deadline δ_i needs to be moved from a source s_i to its destination storage element t_i, the scheduler calls for a data dissemination algorithm that computes a set of paths P_i and bandwidth values, and the computed bandwidth values on all links along these paths are reserved before the start of data transfer between the start and end times of transmission. Similar data dissemination mechanisms based on advance reservation can be found in numerous studies in the literature, e.g., [10, 41, 46].

From the network point of view, it is possible to provide such a guaranteed service by means of having IntServ [6, 7] or MPLS [33] together with RSVP [8], which is recently surveyed in detail by [47]. The next generation networks in [12], on the other hand, naturally enable grid applications to reserve the network bandwidth in advance to guarantee its availability. Furthermore, it should be noted that the guaranteed service of IntServ [7] achieves zero packet loss as long as the related flows stay within their specified traffic parameters, and a firm delay guarantees that a datagram will arrive no later than a certain time after it was transmitted by its source. There are also efforts to provide a zero packet loss network infrastructure to TCP so that it can achieve a sustained high throughput in the science networks [18].

For the best-effort data dissemination problems studied in the literature, a variety of goals are attained. Some best-effort goal examples include the completion of data transfer requests as early as possible, minimizing a cost function associated with the data transfers or maximizing over all link utilization. Since this work focuses on the real-time data transfers, the goal differs from the best-effort problems, and Definition 5 formally gives it.

Definition 5. Given a networked system $G = (V, E)$ and a set of real-time data transfer requests R, the real-time splittable data dissemination problem (RTS/DDP) seeks to maximize the number of satisfied requests where each requested data item is required to be transferred over at most k disjoint paths.

$$RTS/DDP : \max \sum_{r_i \in R} \sum_{p_j \in P_i} f_{ij} x_{ij} \tag{2}$$

$$\sum_{p_j \in P_i} f_{ij} = 1, \forall\ r_i \in R \tag{3}$$

$$\sum_{p_j \in P_i} x_{ij} \leq k, \forall\ r_i \in R \tag{4}$$

$$\sum_{r_i \in R} \sum_{p_j \in P_i} \xi_{ij}^{l} \pi_{ij} f_{ij} \leq c_l, \forall\ e_l \in E \tag{5}$$

$$\sum_{e_l \in p_j} d_l \leq \delta_i, \forall\ r_i \in R\ and\ \forall\ p_j \in P_i\ if\ x_{ij} = 1 \tag{6}$$

$$f_{ij} \leq x_{ij}, \forall\ r_i \in R\ and\ p_j \in P_i \tag{7}$$

$$x_{ij} \in \{0, 1\},\ 0 \leq f_{ij} \leq 1, \forall\ r_i \in R\ and\ p_j \in P_i \tag{8}$$

where x_{ij} is 1 if request $r_i \in R$ is transferred over $p_j \in P_i$, and 0 otherwise; f_{ij} defines the fraction of request $r_i \in R$ that will be transmitted over path $p_j \in P_i$; ξ_{ij}^l is 1 if link $e_l \in E$ is on the path $p_j \in P_i$, and 0 otherwise.

In RTS/DDP, the objective function (2) is to maximize the number of satisfied requests. For any request $r_i \in R$, in (3), the total fraction of scheduled data item must be equal to one in order to guarantee the satisfaction of all requests. For any request $r_i \in R$, in (4), the number of paths that are used to make it satisfiable must be less than or equal to k. For all links $e_l \in E$, in (5), their capacity must not be violated due to all scheduled data transfers. In (6), the delay guarantee associated to a given request must be satisfied on its candidate path if and only if this path is activated. In (7), the indicator variables x_{ij} are link to the f_{ij} variables. Finally, bounds on all variables are introduced in (8).

Theorem 1. RTS/DDP is NP-hard.

Proof. It was shown in [39] that the single source unsplittable flow problem can be reduced to the k-path routing problem with single commodity, which is equivalent to RTS/DDP for $|R| = 1$ and $d_l = 0$ for all $e_l \in E$. Since the single source unsplittable flow problem is known to be NP-hard, RTS/DDP is also NP-hard. □

In this study, RTS/DDP is split into two sub-problems, namely real-time k-path selection problem (RTPS-k) and real-time flow sharing (RTFS), whose solutions are sought in two separate phases. While RTS/DDP is solved, in the first phase, a heuristic algorithm produces a solution to RTPS-k, in which feasible paths for each request are included, if possible. However, having feasible paths for each request does not usually imply that all requests can be simultaneously scheduled along their respective feasible paths because of the link capacity constraints. In such a case, a small number of requests must be rejected for the sake of the remaining ones, which now can be satisfied along their pre-computed paths. In this study, deciding about which requests should stay and which ones should be rejected is handled in the second phase. Thus, another heuristic algorithm takes all the paths generated in the first phase as an input and produces a solution to RTFS, in which a subset of input paths with the minimum cardinality is rejected. Furthermore, it is the duty of the second phase to determine the amount of flow to be directed through each route for the requests that will be satisfied. In the following sections, several algorithms for the solution of RTPS-k and RTFS are introduced.

4 REAL-TIME K-PATH SELECTION

In order to solve RTS/DDP, k paths for each request need to be determined. Each path can be found using any path selection algorithm. In this study, shortest delay based path selection algorithms are employed. Two heuristic algorithms are proposed for RTPS-k in this study: k-shortest path, k-disjoint path.

4.1 k-Shortest Path (kSP)

As shown in Figure 1, in kSP algorithm, k paths for each request is found as follows: Initially, request $r \in R$ is defined by destination machine t_r, data item f_r, and deadline δ_r. If all link delays are assumed to be zero, it is sufficient to reserve a bandwidth value of $r.NoDelayRequiredBandwidth = |f_r|/\delta_r$ on every link along path $p_j \in P_r$ to satisfy the deadline of request. Any link that cannot provide $r.NoDelayRequiredBandwidth$ is first removed from G to form a new undirected graph $tempG$. Then, the Dijkstra's shortest path algorithm is run on $tempG$ in order to find the shortest path from destination machine $r.Destination$ (t_r) to every other storage machine. It is possible that one or more storage machines have data item f_r. Thus, $r.Source$ is chosen to be the storage machine with data item f_r that is the closest to $r.Destination$, and $path_1 \in P_r$ is found. During the second for-loop, starting from $path_1$, remaining k-1 paths are generated. In the i^{th} iteration of this for-loop, a new path is identified to be the shortest one among all paths that are deviations from $path_i$, where all deviations from $path_i$ are found as follows, which is inspired from [48]:

1. Let n be the number of links in $path_i$.
2. From $j = 1$ to n, repeat the following:
3. Delete the j^{th} link of the i^{th} shortest path. Run the Dijkstra's shortest path algorithm to find the shortest path with remaining links. If a path is found, add this path as a possible deviation. Reinstate the deleted link.

All paths found are stored in $r.Paths$. Finally, kSP algorithm returns kSP_List that includes all related information for the second phase. The Dijkstra's shortest path algorithm runs in $O(|V|^2)$. In the worst case, $path_i$ has $|E|$ links. Therefore, all possible deviations from a path are found in $O(|V|^2|E|)$. Hence, kSP algorithm works in $O(k|V|^2|E|R)$.

4.2 k-Disjoint Path (kDP)

The paths that are determined by kSP may include some links more than the other. In order to balance the link utilizations, k-Disjoint Path (kDP) algorithm, which is shown in Figure 2 and inspired from [49], is proposed. Note that kDP algorithm is the same as kSP algorithm until the second for-loop. In the second for-loop, starting from $path_1$, remaining k-1 paths in kDP are produced in a different manner.

kSP algorithm
//Input: $G(V, E)$, request set R, number of paths for each request k
//Output: kSP_List
$kSP\_List = \emptyset$;
for each request $r \in R$
$r.Paths = \emptyset$;
$r.NoDelayRequiredBandwidth = |f_r|/\delta_r$;
$tempG = G - \{e_i : e_i \in E; \textit{and } r.NoDelayRequiredBandwidth > c_i\}$;
Run Dijkstra's algorithm on $tempG$ while $r.Destination$ is source;
$r.Source$ = nearest data storage machine with f_r;
$path_1$ = shortest path between $(r.Source, r.Destination)$ on $tempG$;
$r.Paths = r.Paths \cup path_1$;
for $i = 1$: $k - 1$
$B = \{path_j : a \textit{ deviation from } path_i\}$;
$path_{i+1}$ = Dijkstra's shortest path in set B;
$r.Paths = r.Paths \cup path_{i+1}$;
end for
$kSP\_List = kSP\_List \cup \langle r, r.Paths\rangle$;
end for
return kSP_List;
end algorithm

Figure 1. kSP algorithm

Specifically, in the i^{th} iteration, a new path is determined as follows: Links on $path_i$ are removed from the current network topology $tempG$. After the removal, the next shortest distance path is determined by running the Dijkstra's shortest path algorithm on $tempG$. The procedure is repeated until up to k paths are determined. In order to find k disjoint paths, the Dijkstra's shortest path algorithm is called k times, which results in $O(k|V|^2)$. Hence, kDP algorithm has the time complexity of $O(k|V|^2R)$.

5 REAL-TIME FLOW SHARING (RTFS)

After finding k paths for each request using any one of RTPS-k algorithms, the problem of real-time flow sharing attempts to find out the amount of flow to be delivered through each path in order to maximize the number of satisfiable requests. For the solution of RTFS, two algorithms, namely equal share multi path and balanced share multi path, are proposed.

5.1 Equal Share Multi Path (ESMP)

As shown in Figure 3, equal share multi path algorithm accepts *kSP_List* or *kDP_List* as an input. For each data transfer request in *kSP_List* or *kDP_List*, ESMP first

kDP algorithm
//Input: $G(V, E)$, request set R, number of paths for each request k
//Output: kDP_List
$kDP\_List = \emptyset$;
for each request $r \in R$
$r.Paths = \emptyset$;
$r.NoDelayRequiredBandwidth = |f_r|/\delta_r$;
$tempG = G - \{e_i : e_i \in E; \textit{and } r.NoDelayRequiredBandwidth > c_i\}$;
Run Dijkstra's algorithm on $tempG$ while $r.Destination$ is source;
$r.Source$ = nearest data storage machine with f_r;
$path_1$ = shortest path between $(r.Source, r.Destination)$ on $tempG$;
$r.Paths = r.Paths \cup path_1$;
for $i = 1$: $k - 1$
$tempG = tempG - \{e_j : e_j \in path_i\}$;
$path_{i+1}$ = shortest path between $(r.Source, r.Destination)$ on $tempG$;
$r.Paths = r.Paths \cup path_{i+1}$;
end for
$kDP\_List = kDP\_List \cup \langle r, r.Paths\rangle$;
end for
return kDP_List;
end algorithm

Figure 2. kDP algorithm

tries to schedule request r through single route $path_1$, where $r.RequiredBandwidth_1 = |f_r|/(\delta_r - totalDelay(path_1))$. If all links along $path_1$ have available bandwidth that is at least $r.RequiredBandwidth_1$, request r is considered to be *satisfiable* and link bandwidths are updated accordingly. If it is not possible to schedule a request through single path, ESMP algorithm tries to schedule request r through two different routes $path_1$ and $path_2$, each of which delivers half of the size of request and $r.RequiredBandwidth_1 = (|f_r|/2)/(\delta_r - totalDelay(path_1))$ and $r.RequiredBandwidth_2 = (|f_r|/2)/(\delta_r - totalDelay(path_2))$. If all links along $path_1$ and $path_2$ have available bandwidth that is at least $r.RequiredBandwidth_1$ and $r.RequiredBandwidth_2$, respectively, request r is considered to be *satisfiable* and link bandwidths are updated accordingly. If it is not possible to schedule the request through two paths, the procedure repeats up to k paths at most. If request r cannot be satisfied with k paths, it is labeled as *unsatisfiable*.

In ESMP algorithm, while-loop repeats k times at the worst case for each request. During the i^{th} iteration of while-loop, for-loop inside while-loop costs $O(i|E|)$, where a path can have at most $|E|$ links. As a result, while-loop has a cost of $O(k^2|E|)$ and ESMP algorithm has the time complexity of $O(k^2|E|R)$.

```
ESMP algorithm
  //Input: G(V, E), List = kSP_List or kDP_List, k
  //Output: SatisfiableRequestList
  SatisfiableRequestList = ∅;
  e.Used = 0 for all e ∈ E;
  for each request r ∈ List
    r.Paths = List_r.Paths;
    i = 1; r.State = unsatisfiable;
    while (i ≤ k and r.State = unsatisfiable)
      r.State = satisfiable;
      e.UsedCopy = e.Used for all e ∈ E;
      for j = 1: i
        path = r.Paths_j; myshare = |f_r|/i;
        r.RequiredBandwidth_j = myshare/(δ_r − totalDelay(path));
        for each link e ∈ path
          if (e.UsedCopy + r.RequiredBandwidth_j > c_e)
            r.State = unsatisfiable;
            i = i + 1; break;
          else
            e.UsedCopy = e.UsedCopy + r.RequiredBandwidth_j;
         end if
        end for
        if (r.State = unsatisfiable) break; end if
      end for
    end while
    if (r.State = satisfiable)
      for j = 1: i
        path = r.Paths_j;
        for each link e ∈ path
          e.Used = e.Used + r.RequiredBandwidth_j;
        end for
        SatisfiableRequestList ∪= ⟨r, path, r.RequiredBandwidth_j⟩;
      end for
    end if
  end for
  return SatisfiableRequestList;
end algorithm
```

Figure 3. ESMP algorithm

```
BSMP algorithm
  //Input: G(V, E), List = kSP_List or kDP_List, k
  //Output: SatisfiableRequestList
  SatisfiableRequestList = ∅; e.Used = 0 for all e ∈ E;
  for each request r ∈ List
    r.Paths = List_r.Paths; i = 1; r.State = unsatisfiable;
    r.TotalUsefulBandwidth = 0;
    while (i ≤ k and r.State = unsatisfiable)
      r.State = satisfiable; path = r.Paths_i; r.UsefulBandwidth_i = ∞;
      for each link e ∈ path
        if (r.UsefulBandwith_i > (c_e − e.Used))
          r.UsefulBandwith_i = c_e − e.Used;
        end if
      end for
      r.TotalUsefulBandwidth += r.UsefulBandwidth_i;
      e.UsedCopy = e.Used for all e ∈ E;
      for j = 1: i
        path = r.Paths_j;
        myshare = |f_r| × r.UsefulBandwidth_j/r.TotalUsefulBandwidth;
        r.RequiredBandwidth_j = myshare/(δ_r − totalDelay(path));
       for each link e ∈ path
         if (e.UsedCopy + r.RequiredBandwidth_j > c_e)
           r.State = unsatisfiable;
           i = i + 1; break;
          else
           e.UsedCopy = e.UsedCopy + r.RequiredBandwidth_j;
         end if
        end for
        if (r.State = unsatisfiable) break; end if
      end for
    end while
    if (r.State = satisfiable)
      for j = 1: i
        path = r.Paths_j;
        for each link e ∈ path
          e.Used = e.Used + r.RequiredBandwidth_j;
        end for
        SatisfiableRequestList ∪= ⟨r, path, r.RequiredBandwidth_j⟩;
      end for
    end if
  end for
  return SatisfiableRequestList;
end algorithm
```

Figure 4. BSMP algorithm

5.2 Balanced Share Multi Path (BSMP)

ESMP algorithm attempts to deliver roughly equal amount of flow from each path. However, assigning more flow to paths with more usable bandwidth may increase efficiency. In balanced share multi path algorithm, which is shown in Figure 4, the main difference from ESMP is the computation of *myshare*. In ESMP, $myshare = |f_r|/i$ is proportional to the number of different paths used for request r and all paths carry an equal amount (*myshare*) of data. In BSMP, on the other hand, $myshare = |f_r| \times r.UsefulBandwidth_j/r.TotalUsefulBandwidth$ depends on not only the number of different paths used for request r, but also the available minimum bandwidths along these paths. As a result, different paths usually carry different amounts of data, which is proportional to their useful bandwidth values. BSMP algorithm has the same time complexity of $O(k^2|E|R)$ as ESMP.

6 EXPERIMENTAL RESULTS

In order to gather experimental results, a data grid system simulation platform in [50], namely DGridSim, has been used. DGridSim creates a new data grid system based on the simulation parameters in its GUI when a simulation is started.

DGridSim creates a data grid system that will be composed of sites with zero or more computing elements, one data storage element, and star connected local area network. This local network connects the gateway router and computing/storage elements within site by means of dedicated links, where gateway-computing element links have bandwidth of U $\sim$ [Min CE Link Bandwidth = 400, Max CE Link Bandwidth = 600] MBytes/s; gateway-storage element links have bandwidth of U $\sim$ [Min SE Link Bandwidth = 800, Max SE Link Bandwidth = 1 200] MBytes/s; all in-site links have delay of U $\sim$ [Min Site Link Delay = 0.008, Max Site Link Delay = 0.0012] sec. Note that U $\sim$ means uniformly distributed. The gateway routers are interconnected by a randomly generated internet network topology composed of U$\sim$[Min Routers = 8, Max Routers = 12] edge/core routers and U$\sim$[Min Internet Links = 16, Max Internet Links = 24] links whose bandwidths are U$\sim$[Min Internet Link Bandwidth = 120, Max Internet Link Bandwidth = 180] MBytes/s and delays are U $\sim$[Min Internet Link Delay = 0.0025, Max Internet Link Delay = 0.0075] sec. Based on the data organization model chosen, DGridSim either creates a federative data grid system or hierarchical data grid system as follows. Table 1 summarizes the simulation parameters related to the data grid network.

Data grid systems with federative data organization: DGridSim creates a data grid system of Site Count = 20 sites, each of which includes U $\sim$ [Min Computing Element = 24, Max Computing Element = 36] heterogeneous computing elements and a single U $\sim$ [Min Storage Element = 1, Max Storage Element = 1] storage element. The computing elements have MIPS rating of U $\sim$ [Min CE Capacity = 8 000, Max CE Capacity = 12 000] and storage el-

Parameter	Min	Max
CE Link Bandwidth (MBytes/s)	400	600
SE Link Bandwidth (MBytes/s)	800	1 200
Site Link Delay (sec)	0.008	0.0012
Routers	8	12
Internet Links	16	24
Internet Link Bandwidth (MBytes/s)	120	180
Internet Link Delay (sec)	0.025	0.0075

Table 1. Simulation parameters related to the data grid network

ements have storage capacity of U $\sim$ [Min SE Capacity = 80 000, Max SE Capacity = 120 000] MBytes.

Data Grid systems with hierarchical data organization: DGridSim forms a data grid system of Site Count = 25 sites, which correspond to one Tier-0 site, Tier-1 Site Count = 4 Tier-1 sites, and (25-1-Tier-1 Site Count) = 20 Tier-2 sites. Every Tier-1 site is equipped with a single storage element only whose storage capacity is U $\sim$ [Min SE Capacity = 200 000, Max SE Capacity = 300 000] MBytes. Every Tier-2 site, on the other hand, has U $\sim$ [Min Computing Element = 24, Max Computing Element = 36] heterogeneous computing elements and a single storage element with U $\sim$ [40 000, 60 000] MBytes of storage capacity. Note that minimum/maximum values of the Tier-2 site storage capacity are determined based on the total storage capacity of Tier-1 sites and Tier SE Capacity Ratio. In this study, Tier SE Capacity Ratio = 1 is assumed, which implies that the average value of total Tier-1 storage capacity (250 000 $\times$ 4 = 1 000 000 MBytes) is equal to the average value of total Tier-2 storage capacity (50 000 $\times$ 20 = 1 000 000 MBytes). Table 2 gives the simulation parameters related to the data grid system.

Data Organization	Parameter	Min	Max
Federative	Site Count	20	
	Computing Element	24	36
	CE Capacity (MIPS)	8 000	12 000
	SE Capacity (MBytes)	80 000	120 000
Hierarchical	Tier-0 Site Count	1	
	Tier-1 Site Count	4	
	Tier-2 Site Count	20	
	Tier-1 SE Capacity (MBytes)	200 000	300 000
	Tier-2 Computing Element	24	36
	Tier-2 CE Capacity (MIPS)	8 000	12 000
	Tier-2 SE Capacity (MBytes)	40 000	60 000

Table 2. Simulation parameters related to the data grid system

After the data grid system fabric has been formed, jobs start coming into the system according to Poisson process with Inter-arrival Time = 5 sec. DGridSim will create Job Count = 10 500 jobs whose sizes are U ∼ [Min Job Size = 4.8, Max Job Size = 7.2] MI (Million Instruction); deadlines are U ∼ [Min Deadline = 400, Max Deadline = 600] sec. Furthermore, every job is associated with U ∼ [Min Job Data = 2, Max Job Data = 4] job data among all available job data in the system according to a job data access pattern (Random, Geometric, or Zipf). First 500 of these jobs are created to account for slow start and only last 10 000 jobs are considered in the simulation results. Finally, DGridSim instantiates Data-item Count = 10 000 different job data, each of which has a size of U ∼ [Min DI Size = 800, Max DI Size = 1 200] MBytes. Note that job data are randomly distributed to all sites in the federative data organization, while they all are initially stored in Tier-0 site in the hierarchical model. Table 3 gives the simulation parameters related to the jobs and data-items.

Parameter	**Min**	**Max**
Inter-arrival time (sec)	5	
Job Count	10500	
Job Size (MI)	4 800 000	7 200 000
Deadline (sec)	400	600
Job Data	2	4
Data-item Count	10 000	
DI Size (MBytes)	800	1 200
Data Access Patterns	Random, Geometric, Zipf	

Table 3. Simulation parameters related to the jobs and data-items

The performance of the proposed multi path algorithms are compared against *SP_MinDelay* [50] and *Path Determination* [51] algorithms. According to a very detailed study conducted by [50], among 20 different real-time single path data dissemination algorithms, SP_MinDelay algorithm has shown the best performance under the same simulation scenarios as the ones described herein. So, SP_MinDelay algorithm is expected to provide a competitive lower bound for the multi path algorithms. On the other hand, there is no study similar to [50] that compares the performance of available multi path algorithms in the literature. Consequently, among the multi path routing algorithms surveyed in Section 2, Path Determination algorithm by [51] is chosen due to its proven high performance. This algorithm is based on a variant of the Dijkstra's shortest path algorithm, where the weight of an edge is the ratio of the round-trip time of the path corresponding to the edge to its available bandwidth, and the distance function of a path is computed as the maximum of the weights on each of its constituent edges. As a result, Path Determination algorithm schedules data transfers on those paths that can yield the minimum expected completion time for them.

All simulation results are reported in Tables 4–9, which show the number of satisfied data requests among 10000 submitted requests. In these tables, the performance of the proposed algorithms are presented with respect to k (maximum number of paths for each request), while Path Determination algorithm uses as many paths as necessary to meet the related deadlines due to the virtue of its design, and SP_MinDelay always uses a single path. Furthermore, the tables show mean standard deviations (σ) separately for each algorithm as well.

		k					
Algorithms		2	4	6	8	10	σ
SP_MinDelay		5 231					36
kSP	ESMP	5 621	5 603	5 629	5 623	5 620	38
	BSMP	5 664	5 634	5 616	5 653	5 639	31
kDP	ESMP	5 652	5 599	5 635	5 655	5 645	25
	BSMP	5 660	5 622	5 647	5 623	5 637	34
Path Determination		5 606					25

Table 4. The effect of real-time multi path data dissemination algorithms on the real-time performance of data grid systems that adhere to federative data organization model and random data access pattern

		k					
Algorithms		2	4	6	8	10	σ
SP_MinDelay		9 181					111
kSP	ESMP	9 497	9 489	9 519	9 512	9 512	75
	BSMP	9 486	9 509	9 428	9 504	9 504	82
kDP	ESMP	9521	9 492	9 547	9 451	9 451	101
	BSMP	9 548	9 502	9 599	9 515	9 515	90
Path Determination		9 550					57

Table 5. The effect of real-time multi path data dissemination algorithms on the real-time performance of data grid systems that adhere to federative data organization model and geometric data access pattern

Tables 4, 5, and 6 show the results when the data organization model is federative. According to these tables, kSP and kDP perform very similar to each other, while BSMP is usually superior to ESMP for a particular RTPS-k algorithm. In addition, kSP/BSMP achieves a performance improvement over SP_MinDelay between 4 % and 8 %, which is the highest among the proposed multi path algorithms. As compared to Path Determination, however, kSP/BSMP performs slightly better in most of the simulation scenarios.

Tables 7, 8, and 9 depict the results when the data organization model is hierarchical. Similar to the previous results, kSP achieves a very similar performance to kDP, and BSMP is generally better than ESMP. Furthermore, both kSP/BSMP

Algorithms		k					σ
		2	4	6	8	10	
SP_MinDelay		9 113					150
kSP	ESMP	9 405	9 413	9 164	9 491	9 526	95
	BSMP	9 594	9 351	9 517	9 621	9 280	81
kDP	ESMP	9 424	9 545	9 331	9 381	9 456	78
	BSMP	9 419	9 285	9 291	9 513	9 380	126
Path Determination		9 508					218

Table 6. The effect of real-time multi path data dissemination algorithms on the real-time performance of data grid systems that adhere to federative data organization model and zipf data access pattern

Algorithms		k					σ
		2	4	6	8	10	
SP_MinDelay		4 695					21
kSP	ESMP	5 080	5 101	5 088	5 081	5 096	22
	BSMP	5 104	5 095	5 091	5 092	5 099	15
kDP	ESMP	5112	5 108	5 106	5 097	5 102	19
	BSMP	5 100	5 095	5 089	5 080	5 102	19
Path Determination		5 107					29

Table 7. The effect of real-time multi path data dissemination algorithms on the real-time performance of data grid systems that adhere to hierarchical data organization model and random data access pattern

and kDP/BSMP attain a performance improvement over SP_MinDelay about 8 %, and they perform marginally superior to Path Determination in several simulation scenarios.

Even though the proposed multi path algorithms do not considerably improve on Path Determination algorithm, they achieve comparable or better performance

Algorithms		k					σ
		2	4	6	8	10	
SP_MinDelay		5 425					61
kSP	ESMP	5 798	5 817	5 786	5 905	5 777	138
	BSMP	5 859	5 890	5 812	5 873	5 802	127
kDP	ESMP	5 806	5 817	5 808	5 856	5 811	91
	BSMP	5 884	5 877	5 804	5 890	5 782	114
Path Determination		5 848					147

Table 8. The effect of real-time multi path data dissemination algorithms on the real-time performance of data grid systems that adhere to hierarchical data organization model and geometric data access pattern

Algorithms		**k** 2	4	6	8	10	σ
SP_MinDelay		5 026					55
kSP	ESMP	5 378	5 423	5 401	5 469	5 449	62
	BSMP	5 446	5 389	5 412	5 389	5 402	52
kDP	ESMP	5 352	5 387	5 425	5 440	5 407	41
	BSMP	5 483	5 453	5 459	5 404	5 411	53
Path Determination		5 428					42

Table 9. The effect of real-time multi path data dissemination algorithms on the real-time performance of data grid systems that adhere to hierarchical data organization model and zipf data access pattern

values by means of a fixed set of paths between every source and destination pair, rather than computing a new set of paths for every data transfer request. Consequently, in networks with a huge number of data requests, it may be beneficial to use these precomputed paths in terms of minimizing the data transfer scheduling delays.

According to these tables, it is evident that multi path is proven to be superior to single path data dissemination as far as the real-time performance is concerned. Yet, increasing k does not always yield a better performance. The performance of a data dissemination algorithm depends heavily on the data organization model of data grid system and the data access pattern of applications. Furthermore, common to both data organization models, all algorithms show the best performance for geometric data access pattern, followed by zipf and random. Finally, it should be emphasized that these results contribute to the literature in several ways, including the impact of k on real-time data transfers and performance of multi path algorithms under different data organization models and data access patterns.

7 CONCLUSIONS

The real-time splittable data dissemination problem (RTS/DDP) that seeks to maximize the number of satisfied data transfer requests where each requested data item is required to be transferred over at most k disjoint paths is an important problem to harness the performance of data grid systems for data intensive, real-time application. Efficient solutions to the RTS/DDP are sought in four different combination of schedules generated by the heuristic algorithms for real-time k-path selection and real-time flow sharing problems. The proposed multi path algorithms are implemented in a data grid system simulator in order to compare their performances against very competitive single path and multi path algorithms. The simulation results show that the multi path algorithms can significantly improve the performance of data grid systems against the single path algorithm. Furthermore, how much performance will be gained depends on both the data organization of data grid system and data access pattern by jobs. Finally, kSP/BSMP and kSP-kDP/BSMP

algorithms have shown the best performance for the federative and hierarchical data organization models, respectively.

The future work includes the development of new real-time k-path selection and real-time flow sharing algorithms, and the analysis of their combinations with respect to other multi path algorithms. Furthermore, a more detailed survey and the performance comparison of multi path data dissemination algorithms for data grid systems will be conducted.

REFERENCES

[1] Venugopal, S.—Buyya, R.—Ramamohanarao, K.: A Taxonomy of Data Grids for Distributed Data Sharing, Management and Processing. ACM Computing Surveys, Vol. 38, 2006.

[2] Ferrari, T.—Gaido, L.: Resources and Services of the EGEE Production Infrastructure. Journal of Grid Computing, Vol. 9, 2011, pp. 119–133.

[3] Johnston, W. E.: The Computing and Data Grid Approach: Infrastructure for Distributed Science Applications. Computing and Informatics, Vol. 21, 2002, pp. 293–319.

[4] Ahrens, J. P.—Hendrickson, B.—Long, G.—Miller, S.—Ross, R.—Williams, D.: Data-Intensive Science in the US DOE: Case Studies and Future Challenges. Computing in Science & Engineering, 2011, pp. 14–23.

[5] Nicholson, C.—Camaron, D. G.—Doyle, A. T.—Millar, A. P.—Stockinger, K.: Dynamic Data Replication in LCG 2008. Concurrency and Computation: Practice and Experience, Vol. 20, 2008, pp. 1259–1271.

[6] RFC 1633 Integrated Services in the Internet Architecture: An Overview. Available on : `http://tools.ietf.org/html/rfc1633`, December 2013.

[7] RFC 2212 Specification of Guaranteed Quality of Service. Available on: `http://tools.ietf.org/html/rfc2212`, December 2013.

[8] RFC 2205 Resource Reservation Protocol (RSVP). Available on: `http://tools.ietf.org/html/rfc2205`, December 2013.

[9] Internet2. Available on: `http://www.internet2.edu`, December 2013.

[10] Guok, C.—Robertson, D.—Thompson, M.—Lee, J.—Tierney, B.—Johnston, W.: Intra and Inter-Domain Circuit Provisioning Using the OSCARS Reservation System. International Conference on Broadband Communications, Networks and Systems, October 2006, pp. 1–8.

[11] Rao, N. S. V.—Wing, W. R.—Carter, S. M.—Wu, Q.: Ultrascience Net: Network Testbed for Large-Scale Science Applications. IEEE Communications Magazine, Vol. 43, 2005, No. 11, pp. S12–S17.

[12] Charbonneau, N.—Vokkarane, V. M.—Guok, C.—Monga, I.: Advance Reservation Frameworks in Hybrid IP-WDM Networks. IEEE Communications Magazine, Vol. 49, 2011, No. 5, pp. 132–139.

[13] Chen, S.—Nahrstedt, K.: An Overview of Quality of Service Routing for Next-Generation High Speed Networks: Problems and Solutions. IEEE Network, Vol. 12, 1998, No. 6, pp. 64–79.

[14] Orda, A.: Routing with End to End QoS Guarantees in Broadband Networks. IEEE INFOCOMM, Vol. 1, April 1998, pp. 27–34.

[15] Younis, O.—Fahmy, S.: Constraint-Based Routing in the Internet: Basic Principles and Recent Research. IEEE Communications Surveys & Tutorials, Vol. 5, 2003, No. 1, pp. 2–13.

[16] Xue, G.—Zhang, W.—Tang, J.—Thulasiraman, K.: Polynomial Time Approximation Algorithms for Multi-Constrained QoS Routing. IEEE/ACM Transactions on Networking, Vol. 16, 2008, No. 3, pp. 656–669.

[17] Upadhyaya, S.—Devi, G.: Characterization of QoS Based Routing Algorithms. International Journal of Computer Science & Emerging Technologies, Vol. 1, 2010, No. 3, pp. 133–141.

[18] Dart, E.—Rotman, L.—Tierney, B.—Hester, M.—Zurawski, J.: The Science DMZ: A Network Design Pattern for Data-Intensive Science. IEEE/ACM Annual SuperComputing Conference, November 2013.

[19] Guerin, R.—Orda, A.: Networks with Advance Reservations: The Routing Perspective. IEEE INFOCOMM, Vol. 1, March 2000, pp. 118–127.

[20] Sahni, S.—Rao, N.—Ranka, S.—Li, Y.—Jung, E.-S.—Kamath, N.: Bandwidth Scheduling and Path Computation Algorithms for Connection-Oriented Networks. International Conference on Networking, April 2007, p. 47.

[21] Jung, E.-S.—Li, Y.—Ranka, S.—Sahni, S.: An Evaluation of In-Advance Bandwidth Scheduling Algorithms for Connection-Oriented Networks. International Symposium on Parallel Architectures, Algorithms, and Networks, May 2008, pp. 133–138.

[22] Lin, Y.—Wu, Q.—Rao, N. S. V.—Zhu, M.: On Design of Scheduling Algorithms for Advance Bandwidth Reservation in Dedicated Networks. IEEE INFOCOMM Workshops, April 2008, pp. 1–6.

[23] Theys, M. D.—Tan, M.—Beck, N.—Siegal, H. J.—Jurczyk, M.: A Mathematical Model and Scheduling Heuristic for Satisfying Prioritized Data Requests in an Oversubscribed Communication Network. IEEE Trans. Parallel and Distributed Systems, Vol. 11, 2000, No. 9, pp. 969–988.

[24] Eltayeb, M.—Doğan, A.—Özgüner, F.: A Path Selection-Based Algorithm for Real-Time Data-Staging in Grid Applications. Journal of Parallel and Distributed Computing, Vol. 65, 2005, No. 11, pp. 1318–1328.

[25] Eltayeb, M.—Doğan, A.—Özgüner, F.: Concurrent Scheduling: Efficient Heuristics for Online Large-Scale Data Transfers in Distributed Real-Time Environments. IEEE Transactions on Parallel and Distributed Systems, Vol. 17, 2006, No. 11, pp. 1348–1359.

[26] Naiksatam, S.—Figueira, S: Elastic Reservations for Efficient Bandwidth Utilization in Lambda Grids. Future Generation Computer Systems, Vol. 23, 2007, pp. 1–22.

[27] Chen, B. B.—Primet, P. V.-B.: Scheduling Deadline-Constrained Bulk Data Transfers To Minimize Network Congestion. IEEE International Symposium on Cluster Computing and the Grid, May 2007, pp. 410–417.

[28] Soudan, S.—Chen, B. B.—Primet, P. V.-B.: Flow Scheduling and Endpoint Rate Control in GridNetworks. Future Generation Computer Systems, Vol. 25, 2009, pp. 904–911.

[29] Munir, K.—Cigno, R. L.—Primet, P. V.-B.—Welzl, M.: Planning Data Transfers in Grids: A Multi-Service Queueing Approach. Concurrency and Computation: Practice and Experience, Vol. 24, 2012, No. 4, pp. 405–420.

[30] Munir, K.—Welzl, M.—Pasin, M.—Primet, P. V.-B.: Combining Explicit Admission Control and Congestion Control for Predictable Data Transfers in Grids. Future Generation Computer Systems, Vol. 28, 2012, No. 7, pp. 1121–1132.

[31] RFC 2328 OSPF Version 2. Available on: `http://tools.ietf.org/html/rfc2328`, December 2013.

[32] RFC 1195 Use of OSI IS-IS for Routing in TCP/IP and Dual Environments. Available on: `http://tools.ietf.org/html/rfc1195`, December 2013.

[33] RFC 3031 Multiprotocol Label Switching Architecture. Available on: `http://tools.ietf.org/html/rfc3031`, December 2013.

[34] He, J.—Rexford, J.: Toward Internet-Wide Multipath Routing. IEEE Network, Vol. 22, 2008, No. 2, pp. 16–21.

[35] Lee, Y.—Seok, Y.—Choi, Y.: Traffic Engineering with Constrained Multipath Routing in MPLS Networks. IEICE Transactions on Communication, Vol. E87-B, 2004, No. 5, pp. 1346–1356.

[36] Chen, J.—Chan, S.-H. G.—Li, V. O. K.: Multipath Routing for Video Delivery over Bandwidth-Limited Networks. IEEE Journal on Selected Areas in Communications, Vol. 22, 2004, No. 10, pp. 1920–1932.

[37] Misra, S.—Xue, G.—Yang, D.: Polynomial Time Approximations for Multi-Path Routing with Bandwidth and Delay Constraints. IEEE INFOCOMM, April 2009, pp. 558–566.

[38] Truffot, J.—Duhamel, C.—Mahey, P.: k-Splittable Delay Constrained Routing Problem: A Branch-and-Price Approach. Networks, Vol. 55, 2010, No. 1, pp. 33–45.

[39] Banner, R.—Orda, A.: Multipath Routing Algorithms for Congestion Minimization. IEEE/ACM Transactions on Networking, Vol. 15, 2007, No. 2, pp. 413–424.

[40] Rajah, K.—Ranka, S.—Xia, Y.: Scheduling Bulk File Transfers with Start and End Times. Computer Networks, Vol. 52, 2008, No. 5, pp. 1105–1122.

[41] Li, Y.—Ranka, S.—Sahni, S.: In-Advance Path Reservation for File Transfers in E-Science Applications. The Journal of Supercomputing, Vol. 59, 2012, No. 3, pp. 1167–1187.

[42] Gunter, D.—Kettimuthu, R.—Kissel, E.—Swany, M.—Yi, J.—Zurawski, J.: Exploiting Network Parallelism for Improving Data Transfer Performance. SC Companion: High Performance Computing, Networking, Storage and Analysis, November 2012, pp. 1600–1606.

[43] Dasgupta, M.—Biswas, G. P.: Design of Multi-Path Data Routing Algorithm Based on Network Reliability. Computers and Electrical Engineering, Vol. 38, 2012, pp. 1433–1443.

[44] Gamst, M.—Petersen, B.: Comparing Branch-and-Price Algorithms for the Multi-Commodity k-Splittable Maximum Flow Problem. European Journal of Operational Research, Vol. 217, 2012, pp. 278–286.

[45] Conejero, J.—Tomas, L.—Caminero, B.—Carrion, C.: QoS Provisioning by Meta-Scheduling in Advance within SLA-Based Grid Environments. Computing and Informatics, Vol. 31, 2012, pp. 73–88.

[46] Zhang, W.—Cao, J.—Zhong, Y.—Liu, L.—Wu, C.: Grid Resource Management and Scheduling for Data Streaming Applications. Computing and Informatics, Vol. 29, 2010, pp. 1193–1220.

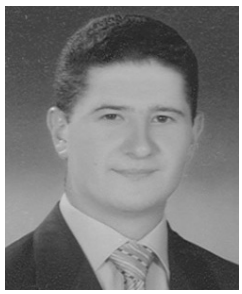
[47] Pana, F.—Put, F.: A Survey on the Evolution of RSVP. IEEE Communications Surveys & Tutorials, Vol. 15, 2013, No. 4, pp. 1859–1887.

[48] Yen, J. Y.: Finding the K Shortest Loopless Paths in a Network. Management Science, Vol. 17, 1971, No. 11, pp. 712–716.

[49] Erlebach, T.: Approximation Algorithms for Edge-Disjoint Paths and Unsplittable Flow. Lecture Notes in Computer Science, Vol. 3484, 2006, pp. 97–134.

[50] Atanak, M. M.: Real-Time Data Management in Data Grid Systems. Anadolu University, Ph.D. Thesis, 2012.

[51] Khanna, G. et al.: Using Overlays for Efficient Data Transfer over Shared Wide-Area Networks. Supercomputing, November 2008, pp. 1–12.



Mustafa Müjdat Atanak received his B.Sc. and M.Sc. degrees in electrical engineering from the University of Southern California in 2002 and 2004, respectively, and his Ph.D. degree in electrical and electronics engineering from the Anadolu University in 2012. He is currently serving as a lecturer in the Department of Electrical and Electronics Engineering, Anadolu University. His current research interests include grid computing, embedded systems, and mobile programming.

Atakan Doğan received his B.Sc. and M.Sc. degrees in electrical and electronics engineering from the Anadolu University in 1995 and 1997, respectively, and his Ph.D. degree in electrical engineering from the Ohio State University in 2001. After receiving his Ph.D. degree, he worked as a post-doctoral researcher at the Ohio State University for nine months. Then he joined the faculty at the Department of Electrical and Electronics Engineering, Anadolu University in 2002, where he is currently Associate Professor and a Vice Chair. Furthermore, he worked as the Deputy Director of the Anadolu University Computer Center for three years. He served as a supervisor or researcher in several national projects, as a researcher in a NFS funded project, and as an advisor for a few national private companies. His current research interests include cloud computing, grid computing, parallel computer architecture, embedded systems, reconfigurable computing, hardware acceleration, and FPGAs.