Cloudlet Seeding: Spatial Deployment for High Performance Tactical Clouds

D. Shires, B. Henz, S. Park, and J. Clarke
U. S. Army Research Laboratory, Computational Sciences Division, APG, MD, USA.

Abstract—Geospatially aware mobile devices, such as smart phones, rely on an architecture that is power constrained and processing power limited. The utility of these devices can be increased by offloading compute-intensive applications to parallel high performance computing (HPC) architectures, thus limiting battery drain, allowing access to large data, and providing faster time to solution. Such a paradigm can be achieved through tactical cloudlets that must operate in environments dominated by mobile ad hoc infrastructure (common in remote environments or military applications). Executing this paradigm is further complicated in that HPC nodes themselves (with some reduced mobility) are now deployable through the use of ruggedized hybrid core technologies. This paper discusses our concept for cloudlet seeding: the static strategic placement of HPC assets in deployed settings in such a way to balance computational load and limit hops to both stationary and mobile HPC nodes.

Keywords: cloudlets, cloud computing, tactical computing, network emulation, high performance computing

1. Introduction

Army soldiers rely on mobile, fast, and up-to-date processing capabilities to be able to complete a myriad of mission assignments and tasks. This is often done in remote areas starved of computing and networking infrastructure or in urban environments where civilian infrastructure operates on differing protocols with security risks. Hence, mobile ad hoc networks (MANETs) will most often provide the operation frameworks for soldiers and military commanders [1]. It is within this operational realm that we are conducting basic and applied research to address how to field High Performance Computing (HPC) capacity within this MANET topology while at the same time increasing capabilities that leverage this new level of computing power. Coupling size, weight, and power with MANET-based operational requirements leads us to the establishment of tactical cloudlets as a way to deliver the required computing power to today’s soldier [2].

Tactical cloudlets couple traditional connotations of cloud-based technology with the overarching goal of maximizing processing power and limiting the time to solution. This is all done in a mobile environment. Cloud computing has a history of focusing predominately on providing computing services with distributed processing and access to data. Commercial processing using high capacity network infrastructure paid little attention to time to solution as a key operational metric, although this is changing a bit with a new-found focus on streaming access speed to data. Military processing requirements push traditional cloud-based processing to new requirements in every dimension. For example, streaming content (such as sensor feeds) to a hand-held mobile smart phone poses numerous problems. First, forcing too much processing through mobile devices will quickly drain power. In a tactical cloudlet the option of piggybacking on other friendly devices may be beneficial. Second, the web of deployed devices must be flexible as signal loss and connectivity maintenance are known problems in deployed environments. Cyber foraging and possible process migration within the tactical cloudlet become critical. These issues are just the start of a long list of differences between commercial cloud systems and deployed tactical cloudlets.

The viability and overall success of tactical cloudlets depends largely on several interwound technical challenges including power-aware computing, cyber foraging (scheduling and process migration), and processing node optimization and characterization. In this paper we put forth another goal that should remain central in the development of tactical cloudlet technology; especially in regards to pushing HPC-level processing to the tactical edge. Tactical cloudlet seeding is an approach that recognizes the importance of spatial proximity for high performance servers to those mobile nodes that they serve. Simply put, cyber foraging will be greatly enhanced if assets are positioned in a way to limit hopping between and promote load balance within districts served by HPC nodes formed by this seeding approach. We discuss the importance of cloudlet seeding as a way to enhance the overall effectiveness of tactical cloudlets. In particular, we describe a methodology to perform cloudlet formation to enhance opportunities for cyber foraging algorithms within a realistic framework for mobile, ad hoc computing devices. This research directly impacts efforts to extend Army cloud computing and the capabilities of analysts and warfighters to the tactical edge.

2. Tactical Cloudlet Seeding

Unlike conventional developed infrastructure, usually consisting of a mix of wired and cellular networks, MANETS are often the only way of ensuring communication between entities deploying to remote areas or to areas with different infrastructure characteristics. Mobile networks are also important to the military mission as they can provide higher security profiles since the network is self-contained.
Deployed devices with the MANETS will be digital, vary in computational power, and operate over networks of varying data rates. Power figures heavily in this deployment as mobile nodes will be drained of battery at varying rates due to connectivity loss, network contention and collisions, computing load, and other operational conditions.

To address this problem, cloudlets have been proposed to address resource poor mobile hardware [3]. The central goal of this paradigm is to leverage computationally powerful resources with infinite power reserves by way of a cloud, but to do so by positioning the resources physically close to the mobile devices to limit the number of hops required for connectivity. That is, reduced latency between mobile device and data center will ensure longer battery life for mobile devices with fast access to information being processed by proximate computing resources.

Carrying this one step further, and by incorporating new heterogeneous and advanced computing methodologies, it is now possible to field near high performance computing (HPC)-level capabilities to soldiers and others in deployed settings. For military planners, HPC modeling and simulation have been useful tools for strategic planning. With tactical HPC using the cloudlet approach, dynamic and real-time processing of live data feeds can be possible for mounted and dismounted soldiers in operational environments. This can have real benefits to improve situational awareness and thus limit fog of war effects during mission execution.

This is a key point in our conceptualization of tactical cloudlet design. By incorporating HPC capacity, along with localized access to the fullest extent possible of mission-critical data at the HPC node, the overall bandwidth allocation dilemma faced by military planners if greatly mitigated. Bandwidth is a scarce commodity in deployed settings using MANETs [4]. Planning and optimizing the distribution is mainly relegated to a strictly strategic exercise subject to lack of complete information, possible exaggerated need from participants, and overall uncertainty. Cloudlet design with HPC-node centralization allows for a greedy-type optimization at the local level that can carry through to allow for an overall optimized network design.

However, access to these powerful devices will be key. While cyber foraging is a way to look for assets to help with processing demand, it does not address spatial locality of HPC to nodes requesting processing service. Tactical cloudlet seeding does just that; it attempts to localize access and balance demand on deployed HPC resources. Done properly, the overall goal is to limit foraging to predefined, seeded HPC districts where the processing load will be sufficient and balanced.

2.1 HPC Node Characteristics

Tactical networks of the future will be built upon a hierarchy of connectivity and capabilities. Hand-held radios and computing devices used by dismounted soldiers will obviously have different profiles and capabilities when compared to those housed in vehicles and stationary encampments. Overall connectivity will depend largely on signal strength, data transfer rates, contention, and positioning of nodes to reduce critical node formation.

Highly mobile compute nodes will always be lagging behind larger, less mobile nodes when it comes to sheer computing power. However, new hybrid architectures are being designed that allow for higher FLOP counts in shrinking footprints. These custom-engineered solutions can be ruggedized and provide a tiered capability for HPC in military applications. More fixed locations, such as Tactical Operations Centers (TOCs), have greater infrastructure and can support elaborate hybrid machines for applications tailored to them. Larger scale traditional multi-core systems, as well as more elaborate custom-engineered solutions like a seven card, water cooled, overclocked GPU-based system shown in Figure 1, can easily be supported in such a facility. Mobile and ruggedized GPU hardware can also easily be supported in platforms such as the High Mobility Multi-purpose Wheeled Vehicles (HMMWV) or certain Unmanned Aerial Vehicles (UAVs).

Fig. 1: Customized GPU-accelerated workstation.

2.2 Network Goals

Tactical cloudlets have numerous measures of success measured largely on security and quality of service (QoS). Numerous techniques have been developed and proposed to enhance, improve, and ensure connectivity of MANETs [5], [6]. HPC nodes within the deployed MANET have special characteristics and specialized goals. While they should ideally posses access to large data rate channels, they should also be configured and distributed in a way that limits the number of hops that might be required of low-power hand-helds to reach them. In this regard, placement becomes critical as these resources should be spatially near assets that they serve. This is the main point of seeding the tactical cloudlets. Done properly, it will ensure speedy turn-around of queries and responses from powerful distributed computing nodes.
3. Problem Formulation

We are given a network based of mobile and fixed connected nodes. For the time being we assume that connections are static and once established are maintained during our period of interest. We are concerned with the connected graph $G = (V, E)$ that represents this network. In practice this will be a geographically constrained subgraph within a larger, distributed network. The vertices, $V$, represent the radios, computing devices, or other electronic systems capable of communicating within the system. These nodes will vary in computing capability, power requirements, etc. The edges, $E$, represent the communication links between these devices. We assume only one link edge between devices. The properties of these edges will vary based on data rates and other network characteristics. These parameters can become quite complex as various parts of this network will consist of mobile ad hoc nodes (MANETs) where geographic and weather effects can impact link capacity due to signal strength, packet loss, etc.

The vertex set, that is the networked components, is the set $V = 1, 2, \ldots, n$. We are concerned with fielding tactical HPC within the cloudlet infrastructure, hence we are actually attempting to determine how to classify the nodes of $V$. In this case, the set $V$ is actually comprised of three disjoint subsets. These include the set of fixed HPC capable nodes, $V_F$, that could reside in deployed settings such as a Tactical Operations Center (TOC), the set $V_H$ of mobile HPC nodes such as those deployable on vehicles, and the set $V_M$ of mobile and/or deployed nodes that are not HPC capable. That is, we have $V$ such that $\forall x \in V, x \in V_F \lor x \in V_H \lor x \in V_M$. For simplicity we assume nodes of overall equal properties within these subsets.

Strategically speaking, and for the static nature of the problem we are currently defining, the initial graph is constructed based on areas where HPC processing capability will be required, where HPC nodes exist in fixed locations, and other areas where dismounted soldier or other sensors and processors will be located. Node locations will therefore be a combination of known locations (such as the fixed nodes) and also areas where some type of coverage is required. Initial node locations will also be based historical or geographical data. In practice, HPC-capable mobile nodes will be positioned in safe areas, be restricted in total number due to cost or other issues, and be within a limited hop range from nodes requesting high rates of compute services. Any node that is not a fixed HPC node has the capability of being assigned to HPC mobile node status. Recognizing the fact that these nodes may reside in areas not amiable to that classification, we also define a set $V_R$ that includes those node numbers that should not be allowed within the set $V_H$. Determining the initial graph configuration, including node placement and edge connectivity, is part of a larger research effort we are undertaking for tactical cloudlet configuration. How this is being performed through network simulation and emulation research will be discussed in a later section.

Our problem, therefore, is to assign every vertex $v$ in $V$ to an HPC node, whether it be mobile or fixed. All elements in $V_M$ are assigned to an element in $V_H$ or $V_F$. Since communication and data transfer happens along these graph edges, the path traversal becomes very important to fielding an ideal system. The weight of any path $p = v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_m$ is the sum of the weights of the edges that it contains:

$$w(p) = \sum_{i=1}^{m} w(v_{i-1}, v_i)$$

The shortest path from $u$ to $v$ is therefore defined as:

$$\delta(u, v) = \min\{w(p) : u \sim^* v\}$$

3.1 Constraints

Since we only have a limited number of HPC-deployed nodes, our goal is to assign every node in our graph to some HPC-computing node. That is, we are attempting to determine a partitioning of the set $V$ into $k$ disjoint subsets $V_1, V_2, \ldots, V_k$ to form a $k$-way partitioning of $V$. In this case $k$ will be equal to $|V_F| + |V_H|$.

Optimal deployment of HPC computing capability to the tactical edge will require the solution of a multi-constrained problem. First, we deal with path traversal or the number of hops a device must travel to reach HPC capable computing nodes. Ideally this number should be as small as possible for several reasons, including amount of data that must be transferred, risks of intermediate node drop-off, etc. Let $u_i$ be the HPC node within partition $i$. We are searching for a total minimum hop count across all $k$ partitions:

$$\min \sum_{i=1}^{k} \sum_{j=1}^{|V_i|} w(u_i, v_j)$$

In certain cases, it may also be necessary to balance the data rates as well as hop counts. Depending on the amount of data stored local on mobile hand-holds versus the HPC node, the impact of these network links will vary considerably. Scalar data based on GPS location, or the transmission of static browser images, will require less total bit transmission with delay and hop counts taking higher priority. We see this as the more common operational case, but are currently in the process of further modifications to equation 3 to better account for those cases where data rates retain a high importance. These cases pose extra difficulties since overall data movement capabilities between two nodes is restricted to the bottleneck link in the path. In the current formulation we only concern ourselves with hop counts or distance from HPC node to the nodes it serves. That is, $\forall e \in E, w(e) = 1$.

Second, we want the overall offered computing load to be balanced within the $k$-way partitions. Doing so will ensure that HPC resources will not be overwhelmed by the nodes that they serve. This computational load is analogous to the partition weight of the $i$th partition, denoted by $w(V_i)$, which is equal to the sum of the weights of the vertices in that partition. Load imbalance then becomes a critical metric and a number that needs to be minimized. In a $k$-way partition, this load imbalance, $LI$, is a ratio of the highest partition weight divided by the average partition weight $[7]$. 

With some generality and assuming an average, distributed computational load, we arrive at $\forall v \in V, w(v) = 1$. An ideal distribution of computing load is hence analogous to seeking approximately equal cardinality of the subsets formed by the $k$-way partitions.

4. Tactical Cloudlet Seeding

Using the constraints developed in the previous section, we describe the approach specified in Algorithm CLOUDLET-SEEDING that can be used to specify the location of HPC mobile nodes within the deployed network. The inputs to the algorithm include the $n \times n$ adjacency matrix $A$ of the graph $G$, the set $V_F$ of fixed HPC nodes, the set $V_R$ of restricted nodes that cannot be designated as mobile HPC nodes, and the scalar $q$ representing the total number of mobile HPC nodes that are allowed.

The algorithm builds the matrix $D[1 \ldots n, 1 \ldots n]$ that gives the shortest path distance between all pairs of vertices. The algorithm also builds and iterates over the power set (set of sets) $B = \{x \in 2^V : (|x| = q) \land (\forall y \in x, y \notin V_F \land y \notin V_R)\}$. The matrix $B$ is therefore all possible $q$-way combinations of the vertices in $V$ excluding the nodes that cannot serve as HPC mobile nodes and those fixed HPC nodes. The number of these sets is given by the binomial coefficient and in this case is $\binom{n-(|V_F|+|V_R|)}{q}$.

In order to hold information about the various partitions, we also define a partition type $p$ on each node in the HPC node in that partition, a scalar keeping count of the number of nodes assigned to that partition, and a list of those nodes assigned to the partition. The variables $p$ and $Z$ are of this type.

The procedure makes use of several ancillary functions. The function INIT-PART returns an initialized array of structures of length $q + |V_F|$ containing information as defined in the $p$-type data structure. At each iteration of the loop, this structure is initialized with the HPC node location data and also the current combination of possible HPC nodes selected from the set $B$. MIN-HOPS is a straightforward iteration down the rows of shortest path matrix where every possible HPC vertex location specified in $p$ is compared against the other for hop counts. In cases where there is a clear winner, the vertex is assigned to that partition immediately. The case where the row happens to correspond to one of the possible HPC locations is a clear example. In this case the hop value will be 0 and that vertex is assigned to that corresponding partition. In cases where there are more than one possible HPC locations with the same minimum hop, the decision is delayed until the function BAL-PART.

At this time, the value of part.count becomes important. Partitions with a lower total weight (number of vertices served) win and are assigned that row. In cases of further ties, we arbitrarily pick the lowest numbered vertex as the winner. The procedure returns a partition structure $Z$ that tells the $q$-way partition assignment of $a[v_1 \ldots v_n]$. Consider the graph as shown in Figure 2 as an input to this algorithm. We assume uniform computational load on all nodes and equal weight on all edges. Nodes 4 and 7 are special; they represent fixed HPC-enabled compute nodes within this graph. All other nodes in the graph represent locations for possible mobile HPC placement (the vector $r$ is empty). In this case we can field one mobile HPC node ($|V_H| = 1$). We start by computing the all-pairs shortest paths to determine hop distance between the nodes. Since HPC nodes under consideration or those fixed will have a hop distance of 0, they will by default be the HPC element of their respective partition. We then generate the possible subsets of size 1 (nodes 1, 2, 3, 5, 6, 8, 9, 10, 11, 12, and 13). Each of these is analyzed according to the criteria in lines 7 and 9. On the first pass, nodes with a clear shortest hop to an HPC node are assigned to that node’s partition. On cases of a tie, the assignment is deferred to the next pass. Here, the weight of both partitions is assessed and the node is assigned to the partition with the lower weight. The weight of this partition is then incremented by that node’s weight.

Consider the graph as shown in Figure 2 as an input to this algorithm. We assume uniform computational load on all nodes and equal weight on all edges. Nodes 4 and 7 are special; they represent fixed HPC-enabled compute nodes within this graph. All other nodes in the graph represent locations for possible mobile HPC placement (the vector $r$ is empty). In this case we can field one mobile HPC node ($|V_H| = 1$). We start by computing the all-pairs shortest paths to determine hop distance between the nodes. Since HPC nodes under consideration or those fixed will have a hop distance of 0, they will by default be the HPC element of their respective partition. We then generate the possible subsets of size 1 (nodes 1, 2, 3, 5, 6, 8, 9, 10, 11, 12, and 13). Each of these is analyzed according to the criteria in lines 7 and 9. On the first pass, nodes with a clear shortest hop to an HPC node are assigned to that node’s partition. On cases of a tie, the assignment is deferred to the next pass. Here, the weight of both partitions is assessed and the node is assigned to the partition with the lower weight. The weight of this partition is then incremented by that node’s weight.
The result of this algorithm is shown in Figure 3. Partition 1 is based at HPC node 4, partition 2 at HPC node 7, and partition 3 at HPC node 11. The solution gives a path no longer than 3 in partition 1 (P1, total hops = 6), no longer than 2 in partition 2 (P2, total hops = 5), and no longer than 2 in partition 3 (P3, total hops = 5). The total hop count is 16 for this partitioning. Partition weights are 4, 4, and 5 for partitions 1, 2, and 3 respectively. The load imbalance in this 3-way partition is 15%.

Fig. 3: An optimal three-way partition of the graph in Figure 2. Nodes 4 and 7 are assisted by a mobile HPC node at location 11.

Contrast this to the worst mobile HPC node assignment discovered by the algorithm which is shown in Figure 4. The mobile HPC node in this case is node 5. \( LI \) is also 15% (not uncommon for such a small problem set), the total hops is 25 (4 for partition P1 serviced by node 4, 10 for partition P2 serviced by node 7, and 11 for partition P3 serviced by node 5). The maximum path length in this case is 4. There is another major problem with this solution. The cloudlet concept of localized access to HPC nodes is severely violated. Several nodes, such as node 1, require connectivity through external cloudlets to connect to their assigned HPC resource. Such a situation will probably be experienced, at least transiently, in a tactical situation. However, strategic solutions should not produce such assignments and routing.

While nodes \( v_1 \) and \( v_m \) within the path \( p = v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_m \) will be members of the same partition set, more rigorous analysis will be needed to verify that nodes \( v_2 \ldots v_m \) will reside in the same partition. In a static analysis, this is not overly important as membership to a partition is not as important as minimizing the overall hops and load imbalance. It may become more important if further refinements for dynamic optimization require members of partitions to move in such a way to maintain connectivity.

4.1 Analysis

The algorithm CLOUDLET-SEEDING is a combination of all-pairs shortest path, brute-force methods for generating the possible subset configurations, and a 2-phase peephole greedy method optimization for resolving partition assignment. The data structures required will generally include simple scalars, vectors of size \( n \), and arrays and structured data types not exceeding sizes of \( n \times n \). Space can also be conserved through the use of more elaborate sparse structures such as linked lists.

We can compute a simple upper bound on the running time of CLOUDLET-SEEDING. Forming the adjacency matrix takes \( O(n^2) \) since there can be at most one edge between any two vertices. Ideal computation of all pairs shortest path will depend on restrictions to edge weights (currently uniform at cost 1). Dijkstra’s algorithm or the Bellman-Ford algorithm will suffice with varying data structures impacting overall performance (binary heap versus Fibonacci heap in the binary-heap implementations). In general though this runtime will be \( O(n^2 \log n) \). These times are all incurred once outside of the main loop.

The generation of the combinations of subsets of size \( q = |V_F| + |V_R| \) becomes more complex and corresponds to the beginning of the main processing loop (iteration counter \( i \)). The function INIT-PART is responsible for generating and initializing partitions for evaluation as HPC nodes. This function will omit nodes already selected as stationary HPC nodes \( V_F \) and nodes that are restricted \( V_R \). This function could generate these subsets by maintaining a list of static pointers to the vertices where these pointers are incremented at each call for the next iteration’s subsets. Of primary concern, however, is the iteration count to generate these combinations. In practice we expect the number of HPC nodes to be a very small percentage of the overall number of fielded binary systems, and we assume the number of fixed HPC nodes and restricted nodes are relatively small. Since the number of HPC nodes within the problem set will be small, we recognize that \( q \) will be some fraction of \( n \). We can then rewrite \( \binom{n}{q} \) as \( \binom{n}{\lambda n} \), where \( 0 \leq \lambda \leq 1 \), with a bounds of \( O(2^{\lambda n} H(\lambda)) \) where \( H \) is the binary entropy function \( H(\lambda) = -\lambda \log \lambda - (1 - \lambda) \log (1 - \lambda) \) [8].

The calls to procedures MIN-HOPS and BAL-PART each must iterate over all rows of the shortest path matrix doing simple minimization comparisons and assignments. Thus
the time for each of these is $O(n(q + |V_F|))$. HOPS and LOAD-BAL simply return the current hop count and nodal balance of the partitions, respectively. The overall runtime of CLOUDLET-SEEDING is thus $O((n(q + |V_F|)2^n H(\alpha))$.

Some trivial optimizations are possible. For instance, should the result of MIN-HOPS yield a result greater than the already established minimum hop count, there is no need to try to balance the partitions. Rather the next iteration of the outer loop can begin immediately.

5. Initial Graph Construction and Optimization

The current static formulation for cloudlet seeding requires some initial configuration of network asset placement with a corresponding assessment as to how the nodes will be linked. The heterogeneous nature of the deployed binary computing devices, from unattended sensors to powerful systems in areas with supporting infrastructure, makes this assessment both tedious and difficult. Often this is done by hand and through empirical data. Locations are based on historical, geographical, meteorological, and sometimes “best guess” estimates based on characteristics of the devices to be deployed.

We are currently designing a system based on network emulation with real-time radio frequency propagation analysis to determine signal loss of devices communicating wirelessly and determine the impact on data transmission to HPC nodes. This will require modifications to the cloudlet seeding approach where communication edges could be weighted to better account for the physical problem.

Furthermore, we are also performing calculations based on line-of-sight (LOS) as a threat assessment tool in an effort to determine areas that are out of harms way. These areas will in most cases be the better candidates for districting HPC resources. Furthermore, with a feedback mechanism built into the network emulator, areas will be chosen such that they provide shelter and enhanced network connectivity. Tactical cloudlet seeding therefore quickly grows into a multifaceted constraint-based optimization problem.

6. Future Work

While this initial assessment attempts to keep the cardinality of the subsets formed roughly equal and limit hops, we recognize that further research is justified as we move forward in our goal to push HPC to the edge. For example, we may wish to specify numerous characteristics to the edges of this connectivity graph. Bandwidth and latency characteristics will differ significantly based on the properties of the nodes (radios versus sensors versus etc.). It should also be noted that the computational capabilities of the nodes in the system will be highly disparate, and the needs of different resources will need to be further developed and represented to the optimization engine. Computational request load will create a dynamic situation where the weights of the nodes will need to be considered. Computing load and demand must also be further defined.

Furthermore, the current strategy serves a strategic goal and is static in nature. We are looking into ways to extend methods to tactical scenarios that are dynamic. Changes on the ground may require mobile HPC assets to be moved. Failure of certain nodes, due to loss of battery or signal loss due to movement, will necessitate a need to recompute HPC-directed network traffic. Compensating for these situations will greatly extend the capabilities of this system.

The computational complexity for optimization problems such as this highlights the need for heuristics especially moving into dynamic real-time adjustments to the deployed network. Accordingly, we plan on investigating other approaches such as those found in multi-level partitioning in graph partitioners to reduce the time complexity of CYBER-DISTRICT [9]. Discovering ways to include parameters such as shortest path distance, along with other parameters deemed important from network science research, into these heuristic-based methods will greatly enhance confidence in an optimally deployed system.

7. Conclusions

Situational awareness and applications demanding at or near real-time processing speeds will continue to push the capabilities of hand-held computing devices. This will continue to be the case even as processor technologies for these devices continue to improve. However, with advanced computing architectures constructed of hybrid accelerators, a new higher level of computing is possible and can be deployed in operational and mobile environments. Accessing these technologies, connected via wired and wireless networks, becomes key as hand-held devices try to off-load processing to both save power and increase computational efficiency.

Tactical cloudlet seeding puts forth an important augmentation to concepts in cyber foraging. Seeking to reduce network node hops in reaching an HPC-enhanced node and balance the workload of these servers, it provides a methodology to greatly increase the abilities of mobile resources in resource-poor areas. We have laid the initial groundwork for this concept, and look forward to expanding the challenging optimization problems that it presents for both static and dynamic, strategic and tactical total network capability.

References


