Optimal Request Clustering for Link Reliability Guarantee in Wireless Networked Control

Yu Chen, Hongwei Zhang
Department of Computer Science, Wayne State University
{yu_chen, hongwei}@wayne.edu

***Abstract*—In wireless networked control systems, ensuring predictable communication link reliabilities among sensors, controllers, and actuators is critical. In such scenarios, different data gathered at the application layer of each sender require different packet delivery ratios (i.e., reliabilities). The lower layers try to accommodate these requests by first mapping each of them into a service level and then deliver the associated data packets to the receiver at the mapped service level. Due to resource constraints and maintenance overhead, the number of supported service levels is usually limited. An important question is then how to determine the set of service levels to maintain and how to map each request to an appropriate service level, such that the requested reliabilities are guaranteed and the total cost of mapping is minimized? We formally formulate this as an optimal request clustering problem since each service level acts as a cluster and can host multiple requests. In particular, we formulate the Migratory Clustering Problem and the Non-Migratory Clustering Problem, depending on whether a request can migrate from one service level to another after its initial assignment. We propose two optimal algorithms to solve both problems.**

I. Introduction

Wireless networks are increasingly being explored for real-time control of physical processes [1] [2]. In wireless networked control systems such as those in industrial automation [3] [4], ensuring predictable communication link reliabilities (usually characterized by packet delivery ratios) among sensors, controllers, and actuators is critical. In such scenarios, the application layer of each sender requests a certain link reliability, and its lower layers try to deliver the associated data packets to the receiver at the requested reliability level. Providing reliability less than what is requested may cause control failures, while providing reliability higher than the actual need could lead to underutilization of communication resources.

To guarantee that packets are delivered at requested reliability levels, Zhang et al. proposed PRKS for wireless networked control [5]. PRKS is a TDMA-based distributed protocol to enable predictable link reliability based on the Physical-Ratio-K (PRK) interference model [6]. Through a control-theoretic approach, PRKS instantiates the PRK model parameters according to in-situ network and environment conditions so that each link meets its reliability requirement. In particular, PRKS defines a conflict graph for a given wireless network: a node in the graph denotes a link with data transfer in the network, and a link exists between two nodes in the graph if the corresponding links in the network interfere with each other according to the PRK model. Under the condition that link reliability is ensured, PRKS schedules as many nodes as possible in the conflict graph.

What remains open is how to guarantee heterogeneous reliability requirements for applications. PRKS has only considered the case when there is one reliability request for each link. In wireless networked control, however, different reliability requests from different applications need to be satisfied. For example, safety-critical information and non-safety-critical information gathered at the sender may well need different reliability guarantees. Ideally, the lower layer should maintain an infinite number of reliability levels to support all types of applications. In practice, however, providing only a (small) set of quantized reliability levels makes sense in many respects. Firstly, it is difficult and costly to support infinite number of reliability levels, especially for resource-constrained embedded systems. Secondly, performance analysis is more tractable with finite number of reliability levels. Then an important question is as follows:

Given a set of reliability requests and the maximum number of service levels that can be maintained, how to determine a set of service levels, and how to map each request to an appropriate service level, such that each request is guaranteed and the total cost of mapping is minimized?

In this paper, we formally formulate the problem as an optimal request clustering problem in the sense that each service level acts as a cluster, while all the requests mapped into a same service level are said to be within a same cluster. Theoretically, a service level may pick any real number between 0 and 1 (i.e., packet delivery ratio is between 0 and 100%), making the problem intractable. Towards addressing the problem, we have made the following contributions:

- We formulate and propose an optimal solution for the Migratory Clustering Problem, where a request can migrate from one service level to another during its lifetime.
- We also formulate and propose an optimal solution for the Non-Migratory Clustering Problem, where a request must stay at the same service level once it is assigned to a certain level.
- Unlike most existing clustering algorithms that just minimize the clustering cost, our algorithms also ensure that

each reliability request is clustered to a service level no less than it, thus the reliability is guaranteed.
- Our clustering algorithms take into account the lifetimes of requests. On the contrary, most existing clustering algorithms did not consider the lifetimes of requests, thus a current optimal clustering may not be optimal in the future due to the expiration of some requests.

II. Related Work

Clustering has been well studied in machine learning under the context of unsupervised learning, where no labels (or cluster centers) are available beforehand [7]. Among all the clustering algorithms, *k*-means [8] and its variants [9] are widely used. The basic idea of *k*-means is to first randomly pick *k* cluster centers; then assign the input points to the closest centers; update the new cluster centers as the means of the points in each cluster. Repeat this procedure until no further improvement can be made. The *k*-means algorithm has the following drawbacks. Firstly, it does not guarantee global optimality due to the random initialization of the cluster centers. A large gap between local optimum and global optimum may exist. Secondly, the Euclidean distance between the point and its cluster center is used to measure the clustering quality, which may lead to unsatisfactory clustering.

Clustering has also been considered in signal processing. Edge detection, in particular, finds a set of edge pixels within an image row (or column) as a problem of optimally partitioning that row into a set of segmentations [10] [11]. An algorithm based on dynamic programming is guaranteed to find the global optimum. The proposed algorithm, however, has the following drawbacks. Firstly, it restricts the search space to a finite set assuming that this will not result in significant resolution loss. In practice, the search space is infinite, thus such assumption is open to justification. Secondly, the proposed algorithm does not consider the lifetime of the input. Assuming data points never disappear after arrival, it takes a one-shot clustering for all the input points. In reality, however, old data points may disappear over time and should not be considered for clustering. For example, in real-time scheduling [14], each data packet is associated with a deadline. It is either transmitted before the deadline or discarded if the deadline has been missed.

III. Migratory and Non-migratory Clustering Problem

A. System model

We consider a transmitting node in a multi-hop wireless control network. The node may generate or forward data to next hop. Assume that time is slotted and indexed by $t \in \{0, 1, 2, ...\}$.

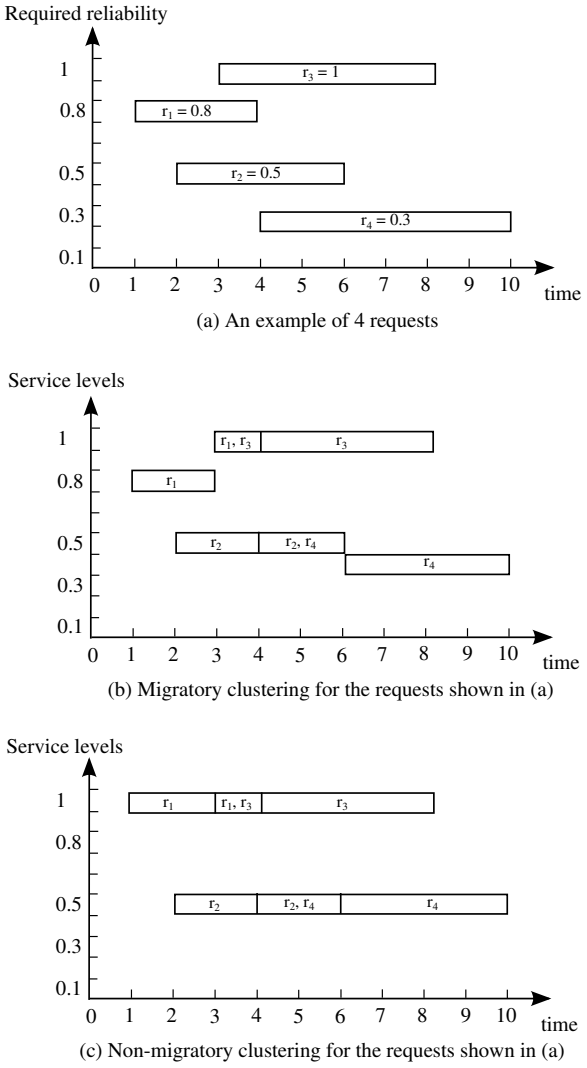
Request. A request $r_i(t)$ is characterized by a triple (r_i, a_i, e_i), where $r_i \in [0, 1]$ is the required reliability, a_i is the time when the request arrives at the node, and e_i is the time when the request expires, respectively. The required reliability stays unchanged during the lifetime of the request, i.e., $r_i(t) = r_i$ for $t \in [a_i, e_i]$. Figure 1(a) shows an example of 4 requests. In the example, request $r_1(t)$ arrives at $t = a_1 = 1$ and expires at $t = e_1 = 4$, requiring a reliability of $r_1 = 0.8$, and so on. Note that given a time instant t, the number of live requests gathered at the node may be less than 4. Unlike bursty data traffic in the regular Internet, data traffic in wireless networked control tend to be periodic (e.g., periodic sensor sampling) and predictable [12]. Thus we assume, as in many existing studies [13] [14], that the request information is known in this study.

Service level. A service level is the actual link reliability provided to a request. In a running system, it usually requires certain memory and computation overhead to maintain a service level [5]. It is therefore reasonable to assume that at any time t, the node can only maintain L service levels. Upon its arrival, each request must be assigned to a service level $s_j(t) \in [0, 1]$ ($j \in \{1, 2, ..., L\}$) immediately. A service level acts as a cluster and can host multiple requests. Without ambiguity, we will use the terms *service level* and *cluster* interchangeably in the paper.

Migratory and non-migratory clustering. We consider two types of clustering: *migratory* and *non-migratory* clustering. The former allows a request to make a transition to another cluster from one time instant to another, while the latter requires a request to stay in the same cluster once assigned. Figure 1(b) shows an example of migratory clustering for the requests shown in Figure 1(a). For instance, request 1 is assigned to service level 0.8 from $t = 1$ to $t = 3$, and assigned to service level 1 from $t = 3$ to $t = 4$. Figure 1(c) shows an example of non-migratory clustering for the same requests, where request 1 must stay at service level 1 during its lifetime due to the non-migratory constraint. In both cases, there are only two service levels maintained at any time. Generally, migratory clustering finds its application in scenarios where migrating a request from one cluster to another introduces negligible overhead. On the other hand, non-migratory clustering is more suitable for cases where migration is expensive or infeasible. Therefore, we define the clustering cost as time-dependent as follows:

$$\underset{r_i(t)\to s_j(t)}{c_i(t)} = \begin{cases} s_j(t) - r_i, & \text{if } s_j(t) \geq r_i; \\ +\infty, & \text{if } s_j(t) < r_i. \end{cases} \tag{1}$$

The notation $r_i(t) \to s_j(t)$ denotes that a request $r_i(t)$ is mapped to a service level $s_j(t)$ at time t. The definition is quite intuitive: if the provided reliability is higher than the request, the cost is the difference between the service level and the request, accounting for the overprovisioning of the resource; if a service level is lower than the request, the cost is infinite, meaning the reliability requirement cannot be met, thus the clustering is not acceptable. The hard constraint on reliability makes sense for wireless networked control since reliability lower than the request may well lead to control divergence or



(a) An example of 4 requests

(b) Migratory clustering for the requests shown in (a)

(c) Non-migratory clustering for the requests shown in (a)

Fig. 1. An illustration of migratory and non-migratory clustering with 4 requests and 2 service levels. At any time t, only 2 service levels can be maintained.

even catastrophe.

B. Problems definition

Migratory Clustering Problem (MCP). Given a set of requests $\mathbf{r}(t) = \{r_1(t), r_2(t), ..., r_N(t)\}$. For any $i \in [1, N]$ and $t \in [a_i, e_i]$, $r_i(t) = r_i$. Let $t_{min} = min\{a_1, ..., a_N\}$ and $t_{max} = max\{e_1, ..., e_N\}$. Denote a clustering $= \{\mathbf{s}(t_{min}), ..., \mathbf{s}(t_{max})\}$, where $\mathbf{s}(t) = \{s_1(t), s_2(t), ...\}$ is a set of service levels at time t, and $||\mathbf{s}(t)|| \leq L, \forall t \in [t_{min}, t_{max}]$. Then the problem is to find a clustering * to

$$Minimize \quad \sum_{i=1}^{N} \sum_{t=a_i}^{e_i} \underset{r_i(t) \to s_j(t)}{c_i(t)}, \tag{2}$$

where the second summation is the total cost of clustering request $r_i(t)$ during its lifetime.

Non-Migratory Clustering Problem (NMCP). Given a set of requests $\mathbf{r}(t) = \{r_1(t), r_2(t), ..., r_N(t)\}$. For any $i \in [1, N]$ and $t \in [a_i, e_i]$, $r_i(t) = r_i$. Let $t_{min} = \min\{a_1, ..., a_N\}$ and $t_{max} = \max\{e_1, ..., e_N\}$. Denote a clustering $= \{\mathbf{s}(t_{min}), ..., \mathbf{s}(t_{max})\}$, where $\mathbf{s}(t) = \{s_1(t), s_2(t), ...\}$ is a set of service levels at time t, and $||\mathbf{s}(t)|| \leq L, \forall t \in [t_{min}, t_{max}]$. Since $r_i(t)$ is mapped to a same $s_j(t)$ during its lifetime $t \in [a_i, e_i]$, the problem is to find a clustering * to

$$Minimize \quad \sum_{i=1}^{N} (e_i - a_i) \times \underset{r_i(t) \to s_j(t)}{c_i(a_i)}. \tag{3}$$

IV. Optimal Algorithm for Migratory Clustering

A. Main idea

At first glance, the problem MCP appears very similar to the *k*-means clustering problem. However, they are fundamentally different. Firstly, the directed distance between a cluster member and the cluster center can be either positive or negative in the *k*-means problem. In our problem, a service level cannot be lower than the requests assigned to it, otherwise an infinite cost would incur. Secondly, *k*-means did not consider the lifetime of the requests, thus a current optimal clustering may not be optimal in the future due to the expiration of some members.

Minimize clustering cost at each time instant. Rewrite (2) into (4), the objective is equivalent to minimize the total cost at each time instant.

$$Minimize \quad \sum_{t=t_{min}}^{t_{max}} \sum_{i=1}^{N} \underset{r_i(t) \to s_j(t)}{c_i(t)} \times I_i(t) \tag{4}$$

$$I_i(t) = \begin{cases} 1, & \text{if } t \in [a_i, e_i]; \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The indicator $I_i(t)$ is introduced since a request is not necessarily alive during the whole period of $[t_{min}, t_{max}]$. A request is only assigned to a service level if it is still alive.

In the following, we further discuss some observations on how to minimize the total cost at each time t. In particular, we first prove that we can narrow down the search space into a finite set without performance loss, and then we present a dynamic programming algorithm to find the optimal clustering.

Pruning search space. In theory, the service levels can be any real number in the range $[0, 1]$, making the problem intractable. Fortunately, we can leverage properties of the problem to shrink the search space into a finite set.

Lemma 1. For MCP, at any time t, the reliability service levels must be selected from the set of requests in order to minimize the cost.

Proof. We prove by contradiction. Without loss of generality, let $\widetilde{\mathbf{r}}(t) = \{r_1(t), r_2(t), ..., r_M(t)\}$ be a set of live requests at time t sorted in non-decreasing order of reliability, and $\mathbf{s}(t) = \{s_1(t)^*, s_2(t)^*, ..., s_K(t)^*\}$ an optimal clustering for $\widetilde{\mathbf{r}}(t)$. Assume there is some $s_j(t)^* \notin \widetilde{\mathbf{r}}(t)$.

Case 1. $s_j(t)^* < r_1(t)$. In this case, if any request is assigned to $s_j(t)^*$, it would introduce a cost of $+\infty$ according to (1), which obviously is not optimal; if no request is assigned to $s_j(t)^*$, this service level would be wasted, meaning that we could have a better clustering by choosing another service level, thus $s_j(t)^*$ is not optimal either.

Case 2. $s_j(t)^* > r_M(t)$. In this case, if we move $s_j(t)^*$ to $r_M(t)$, the cost would be further reduced, meaning $s_j(t)^*$ is not optimal.

Case 3. $r_m(t) < s_j(t)^* < r_n(t)$ for some requests $r_m(t)$ and $r_n(t)$. In this case, similar to Case 2, we could reduce the cost by moving $s_j(t)^*$ to $r_m(t)$. By doing this, the costs for clustering requests lower than $r_m(t)$ would reduce, while the costs for clustering requests higher than $r_n(t)$ remain unchanged. Overall, the total cost would reduce, meaning $s_j(t)^*$ is not optimal. ■

Lemma 2. At any time t, the highest-reliability request from the current live set must be selected as a service level.

Proof. This property is straightforward noting that assigning a service lower that than the request leads to infinite cost. If we do not include a service level that equal to the highest live request at current time, there is no service level that can host this request. ■

Dynamic programming. Inspired by an edge detection algorithm [10], we adopt a dynamic programming subroutine MCP-Slot to find an optimal clustering for the live requests at time t.

B. The algorithms

Given a set of live requests at a time slot, Algorithm 1 optimally clusters these requests into different service levels. In the algorithm, $OPT(n,l)$ denotes the optimal cost for clustering the first n requests using l service levels, as shown in Figure 2. Line 3 of the algorithm shows that the cost is 0 if there is no request alive at current time. Line 5 means the clustering cost is 0 if there is no service level available thus no clustering is performed. Line 6 denotes that only one service level is available for clustering. In this case, the highest request, i.e., r_n, is selected as the service level according to Lemma 2. Lines 7 to 12 compute the optimal cost for $OPT(n,l)$ for other n and l. If $n \leq l$, the cost is 0 since each request can make itself as a service level. Otherwise, dynamic programming is applied. The key idea is, to cluster n requests into l levels, we first find the optimal cost of clustering the first k requests into $l-1$ levels, then cluster the remaining requests, i.e., $\{r_{k+1}, r_{k+2}, ..., r_n\}$, into the l-th level. According to Lemma 2, the l-th level must be r_n. Among all possible choices of k, we find the one leading to the least cost. This process iterates until N requests are clustered. At last, return the optimal cost of clustering N requests into L levels.

Note that our solution differs from the aforementioned algorithm [10] in several ways. Firstly, rather than restricting the search space into a finite set without justification, we first weaken this assumption and then prune the search space without any performance loss (i.e., Lemma 1). Secondly, the signal processing algorithm uses a cluster-wise fitness function without the notion of cluster center, while we use a request-wise cost function and require that the value of a cluster center is no less than all of its members. Thirdly, the existing algorithm does not specify how to determine $opt(j)$ for $j = 0, 1, ..., n$.

Algorithm 1 MCP-Slot

1: Sort current live requests in non-decreasing order of the required reliability;
2: **for** $l = 0$ to L
3: $\quad OPT(0,l) = 0$;
4: **for** $n = 1$ to N
5: $\quad OPT(n,0) = 0$;
6: $\quad OPT(n,1) = \sum_{i=1}^{n} |r_n - r_i|$;
7: **for** $l = 2$ to L
8: $\quad$ **for** $n = 1$ to N
9: $\quad\quad$ **if** $n \leq l$
10: $\quad\quad\quad OPT(n,l) = 0$;
11: $\quad\quad$ **else**
12: $\quad\quad\quad OPT(n,l) = \min_{l-1 \leq k \leq n-1} \{OPT(k, l-1) + \sum_{i=k+1}^{n} |r_n - r_i|\}$;
13: **return** $OPT(N,L)$

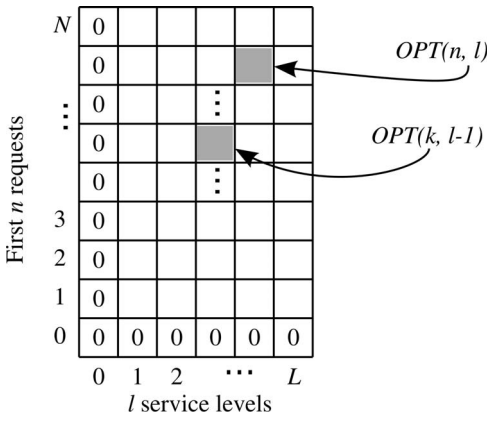


Fig. 2. The table of solutions for $OPT(n,l)$ in Algorithm 1. Fill the table from bottom to top, left to right. When filling in box, we only need to look at boxes we have already filled in.

Then, we can apply Algorithm 1 to generate the clustering for each time instant between t_{min} and t_{max}. We denote this algorithm as Algorithm 2.

C. Correctness and complexity of the algorithms

Given that the Algorithm MCP-Slot is a basic element of Algorithm MCP, we analyze the correctness of MCP-Slot.

Algorithm 2 MCP-Opt

1: t_{min} =min$\{a_i\}$, t_{max} =max$\{e_i\}$;
2: **for** $t = t_{min}$ to t_{max}
3: Current live requests set $\widetilde{\mathbf{r}}(t) = \emptyset$;
4: **for** $i = 1$ to N
5: **if** $a_i \leq t \leq e_i$
6: $\widetilde{\mathbf{r}}(t) = r_i \cup \widetilde{\mathbf{r}}(t)$;
7: Call Algorithm 1 to compute a clustering for $\widetilde{\mathbf{r}}(t)$;

Theorem 1. Given a set of live requests at time t, Algorithm 1 gives an optimal clustering for that time instant.

Proof. For $l = 1$, $OPT(n,1) = \sum_{i=1}^{n} |r_n - r_i|$ is the only way of clustering due to Lemma 2. For $l > 1$, we just need to show that $OPT(n,l) = \min_{l-1 \leq k \leq n-1} \{OPT(k, l-1) + \sum_{i=k+1}^{n} |r_n - r_i|\}$ is actually optimal. To compute $OPT(n,l)$, we only have two choices for service level l:

Case 1. Leave l: This means that the introduction of the l-th service level will not reduce the cost obtained by clustering n requests over $l-1$ levels. Thus, $OPT(n,l) = OPT(n, l-1)$.

Case 2. Take l: Then we have one more service level to use. The best we can do is to assign k requests into $l-1$ levels, and $n-k$ requests into the l-th level. In determining what k is, we exhaustively search a k such that the cost is minimized. In this case, therefore, $OPT(n,l) = \min_{l-1 \leq k \leq n-1} \{OPT(k, l-1) + \sum_{i=k+1}^{n} |r_n - r_i|\}$. Note that the term $\sum_{i=k+1}^{n} |r_n - r_i|$ is due to Lemma 2.

Taking two cases together, we have $OPT(n,l) = min\{\min_{l-1 \leq k \leq n-1} \{OPT(k,l-1) + \sum_{i=k+1}^{n} |r_n - r_i|\}, OPT(n, l-1)\}$. Due to the cost function we use, it is clear that the clustering cost decreases as the number of service levels increases, thus we can ignore Case 1. ■

The complexity of Algorithm 1 is dominated by the loop block from line 7 to line 12. Clearly, it is $O(LN^2)$. Therefore, the complexity of Algorithm 2 is $O(LN^2T)$, where $T = t_{max} - t_{min}$.

V. Optimal Algorithm for Non-migratory Clustering

Unlike MCP, which finds an optimal clustering at each time instant independently, NMCP must determine the service level for each request upon its arrival, and each request stays at the same level during its lifetime.

A. The algorithm

We propose Algorithm 3 to solve the NMCP problem. In the algorithm, $OPT(n,l)$ denotes the optimal cost for clustering the first n requests using l service levels for any possible combinations of n and l. The dynamic programming procedure is similar to Algorithm 1. The major difference lies in the way of mapping a set of requests into a service level. The function MAPPING() maps a set of input requests using one service level. Note that there may be temporal reuse of this service level. Figure 3 illustrates this. In the example, there are 7 requests and only one service level is maintained. The function MAPPING() first partitions the requests into 2 disjoint subsets such that any request from first subset will not overlap with any request from second subset in time. These two subsets therefore can use their own service levels without violating the constraint of only one service level maintained at any time. Within each subset, the highest request is selected as the service level. In practice, the number of disjoint subsets determines the degree of temporal reuse of a service level.

Algorithm 3 NMCP-Opt

1: **for** $l = 1$ to L
2: $OPT(0,l) = 0$;
3: **for** $n = 1$ to N
4: $OPT(n,0) = 0$;
5: $OPT(n,1)$ =MAPPING$([r_1, ..., r_n])$;
6: **for** $l = 2$ to L
7: **for** $n = 1$ to N
8: **if** $n \leq l$
9: $OPT(n,l) = 0$;
10: **else**
11: $OPT(n,l) = \min_{l-1 \leq k \leq n-1} \{OPT(k, l-1) +$ MAPPING$([r_{k+1}, ..., r_N]\}$;
12: **return** $OPT(N,L)$
13: MAPPING(**r**)
14: Partition set **r** into disjoint subsets such that requests from different subsets do not overlap with each other in lifetime;
15: **for** each disjoint subset
16: Make the highest request in the subset as the center;
17: Compute the overall cost of this subset;
18: **return** Total cost of all disjoint subsets;

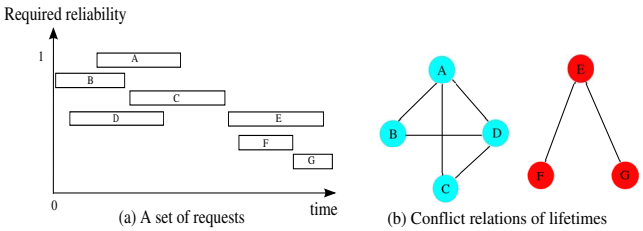


Fig. 3. An illustration of temporal reuse of a service level.

B. Correctness and complexity of the algorithm

Theorem 2. Given a set of requests, Algorithm 3 gives an optimal clustering for NMCP.

Proof. For $l = 1$, $OPT(n,1) =$ MAPPING$([r_1, ..., r_n])$ is the only way of clustering to minimize cost without violating

reliability requirement. For $l > 1$, assume we have computed and stored the optimal clustering for the first $n-1$ requests. To compute $OPT(n, l)$, we assign k requests into $l-1$ levels, and $n-k$ requests into the l-th level. In determining what k is, we exhaustively search a k such that the cost is minimized. We consider the following two cases:

Case 1. There is no overlapping in lifetime between any two of the requests in $[r_{k+1}, ..., r_n]$. In this case, the function MAPPING() partitions the requests into $n-k$ disjoint subsets (i.e., each request make itself a cluster). Thus the cost is 0, which is optimal.

Case 2. There is overlapping in lifetime between some requests in $[r_{k+1}, ..., r_n]$. Without loss of generality, suppose the $n-k$ requests are partitioned into m disjoint subsets. Due to temporal reuse of one service level and exhaustive search of k, the clustering is optimal. ■

The function MAPPING() can be easily implemented in $O(N^2)$, thus the complexity of Algorithm 3 is $O(LN^3)$.

VI. Simulation Results

In this section, we compare the proposed optimal algorithms with a variant of the well known *k*-means clustering via MATLAB simulations. The *k*-means clustering does not guarantee reliability by default. To allow it comparable with our algorithms, we modify it by making the service level of each cluster be the highest request in the cluster. We also use M-*k*-means and N-*k*-means to denote migratory and non-migratory *k*-means clustering, respectively. Figure 4 shows the total cost of different methods when 100 requests and 100 time slots are simulated. The reliability of each request is randomly generated from $[0, 1]$, and the arrival time and expiration time are randomly picked from $[1, 100]$. The result shows that both MCP-Opt and NMCP-Opt outperform their counterparts M-*k*-means and N-*k*-means. The reason is because *k*-means clustering randomly initializes the cluster centers and does not guarantee global optimality by design. In addition, MCP-Opt outperforms NMCP-Opt since MCP can optimize the clustering on a time slot basis while NMCP does not have such degree of freedom. In all cases, the cost decreases as the number of service levels increases. However, the performance gain becomes negligible when the service level reaches certain threshold. Figure 5 shows the total cost of different methods when 20 service levels and 100 time slots are used. As the number of requests increases, the cost also increases in all cases.

VII. Conclusion

We have propose two algorithms to address the problem of clustering heterogeneous reliability requirements into a limit set of service levels. Our solutions are optimal, and they also provide guaranteed reliability, which is critical for wireless networked control. Future work includes incorporating our algorithms into a running system by considering the wireless channel and medium access control [5].

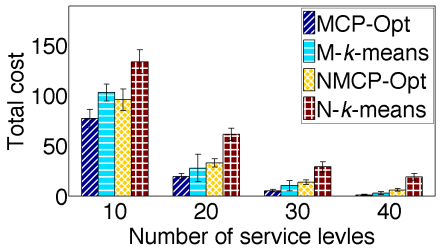


Fig. 4. Total cost vs. number of service levels L (number of requests $N = 100$)

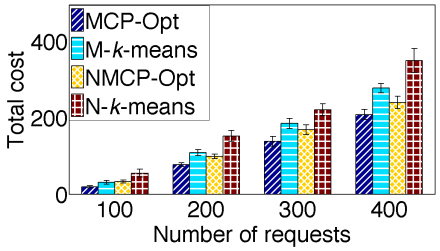


Fig. 5. Total cost vs. number of requests N (number of service levels $L = 20$)

References

[1] J. Baillieul and P. J. Antsaklis. Control and communication challenges in networked real-time systems. *Proceedings of the IEEE*, 95(1):9-28, 2007.

[2] M. Pajic, S. Sundaram, G. J. Pappas, and R. Mangharam. The wireless control network: a new approach for control over networks. *IEEE Transactions on Automatic Control*, 56(10):2305-2318, 2011.

[3] S. Han, X. Zhu, A. K. Mok, D. Chen, M. Lucas, and M. Nixon. Reliable and real-time communication in industrial wireless mesh networks. in *IEEE RTAS*, 2011.

[4] A. Willig. Recent and emerging topics in wireless industrial communications: A selection. *IEEE Transactions on Industrial Informatics*, 4(2):102-124, 2008.

[5] H. Zhang, X. Liu, C. Li, Y. Chen, X. Che, F. Lin, L. Y. Wang, and G. Yin. Scheduling with predictable link reliability for wireless networked control. In *IEEE IWQoS*, 2015.

[6] H. Zhang, X. Che, X. Liu, and X. Ju. Adaptive instantiation of the protocol interference model in wireless networked sensing and control. *ACM Transactions on Sensor Networks*, 10(2), 2014.

[7] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515-528, 2003.

[8] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129-137, 1982.

[9] D. Arthur and S. Vassilvitskii. *k*-means++: the advantages of careful seeding. In *ACM SODA*, 2007.

[10] B. Jackson, J. D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. Sangtrakulcharoen, L. Tan, and T. T. Tsai. An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters*, 12(2):105-8, 2005.

[11] J. D. Scargle and M. K. Quweider. Edge detection using dynamic optimal partitioning. In *IEEE ICASSP*, 2006.

[12] R. Zurawski. Industrial communication technology handbook. *CRC Press*, 2014.

[13] I. Hou and P. R. Kumar. Scheduling periodic real-time tasks with heterogeneous reward requirements. In *IEEE RTSS*, 2011.

[14] T. Carley, M. A. Ba, R. Barua, and D. B. Stewart. Contention-free periodic message scheduler medium access control in wireless sensor/actuator networks. In *IEEE RTSS*, 2003.