The Resource Manager for the Defense Advanced Research Projects Agency Spectrum Collaboration Challenge Test Bed

Jason W. Mok, Alexander L. Hom, Jason J. Uher, and David M. Coleman

ABSTRACT

A key component of success in the Defense Advanced Research Projects Agency (DARPA) Spectrum Collaboration Challenge (SC2) was ensuring that each competitor had fair access to the limited physical resources available in the competition. The Johns Hopkins University Applied Physics Laboratory (APL) designed and developed a custom Resource Manager as part of the Colosseum, the wireless research test bed at the foundation of the SC2 competition. By allocating resources through a token system, the Resource Manager ensured that competitors had fair and equal access to resources in the Colosseum. The Resource Manager also provided mechanisms for automated experiment handling and orchestration that increased the scheduling efficiency of the resources and gave competitors equal access to all 128 nodes in the Colosseum. From 2016 to 2019, the Resource Manager maintained continuous availability of Colosseum resources that enabled international competitors to develop new artificial intelligence algorithms for radio frequency (RF) spectrum management.

INTRODUCTION

In 2016, the Defense Advanced Research Projects Agency (DARPA) launched the Spectrum Collaboration Challenge (SC2), seeking to motivate novel artificial intelligence (AI) solutions to better manage the oversubscribed radio frequency (RF) spectrum. "In this first-of-its-kind collaborative machine-learning competition, competitors . . . reimagine[d] new spectrum access strategies in which radio networks autonomously collaborate to dynamically determine how the . . . spectrum should be used moment to moment, avoiding interference and jointly exploiting opportunities."[1]

In support of SC2, APL designed and built a wireless research test bed known as the Colosseum. This collection of resources facilitated research in autonomous spectrum management across a set of collaborative intelligent radio networks. The resources included computing resources, software-defined radio (SDR) hardware, an RF Emulation System (see the article by Barcklow et al. in this issue for more on this system), emulated backhaul networks, Internet Protocol (IP) traffic streams representing realistic applications (see the article by Curtis et al. in this issue for more on this system), and an emulated GPS service. The Colosseum provided services for research (e.g., secure data storage) and competition (e.g., scorekeeping). Over the 3 years of the competition (2016–2019), the Colosseum was remotely accessible to more than 100 competitors across 30 teams spanning 5 countries.

Within the test bed, there were 128 standard radio nodes (SRNs), and each was composed of a server and an SDR. The SRNs hosted competitor algorithms and provided interfaces to the Colosseum services. (See the article by White at al. in this issue for details on SRNs.) Even though this collection of SRNs made the Colosseum the world's largest test bed for wireless communication research, by design it limited the amount of time and resources allocated to each competitor for executing experiments so that it could ensure equal access to all competitors. The Colosseum did this via its Resource Manager. The Resource Manager accepted resource requests from the competitors, allocated resources in a manner that provided equal access to all competitors, and synchronized Colosseum services for consistent and repeatable experiments.

RESOURCE MANAGER OVERVIEW

The Resource Manager comprised the Colosseum website, a reservation system, an automated scheduler, and an orchestrator. The Resource Manager's reservation and scheduling process is illustrated in Figure 1. On the Colosseum website, competitors could view available resources and make requests for resources through the reservation system. The reservation system validated requests and apportioned SRNs to competitors during the request time. The competitors were able to request manual control of the SRNs or allow the Resource Manager to schedule the experiment when SRNs became available, orchestrate the experiment automatically, and return results to the competitor's network-attached storage. For each experiment submitted for automated handling, the scheduler would determine an appropriate time to reserve SRNs and the orchestrator would coordinate the Colosseum services for the experiment. The subsequent sections in this article discuss each system in more detail.

THE RESERVATION SYSTEM

Competitors were able to request SRNs on the Colosseum via the Resource Manager website. Reservation requests were public and shared on the Colosseum website across competitors. Each request was in the form of a reservation, which included the competitor container to be allocated on the SRN during the reservation, the type of execution (manual or automated), the number of SRNs, and the duration of the reservation.

To ensure equal access to Colosseum resources, the Resource Manager used a token system for manual reservations. A token represented a finite amount of time for use of one SRN in an experiment. The Resource Manager allocated tokens weekly to competitors, thus guaranteeing equal access to the Colosseum within weekly boundaries. Competitors could select how to use

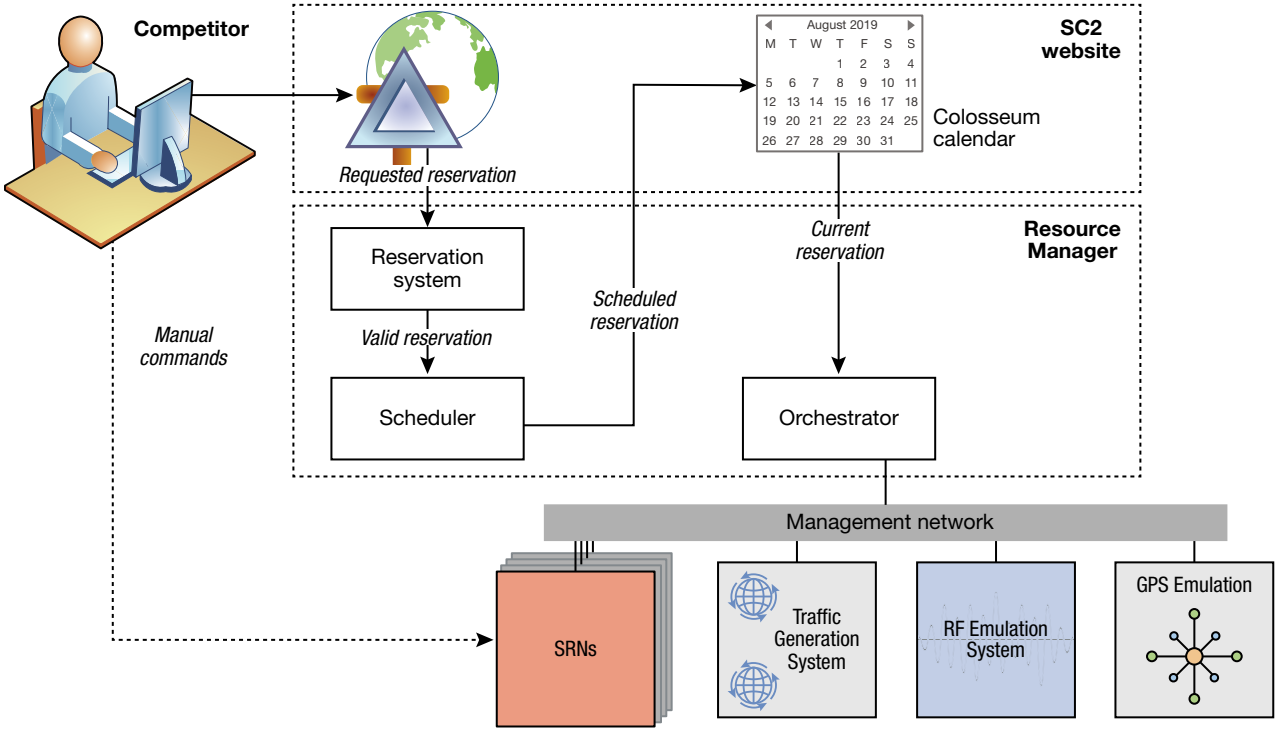


Figure 1. High-level illustration of the Colosseum Resource Manager's reservation and scheduling process. Each competitor could schedule an experiment in the Colosseum through a manual reservation or an automated reservation. The Resource Manager scheduled experiments from the experiment queue as SRNs became available.

their tokens to reserve SRNs. For example, one competitor might have selected five 2-hour experiments with 10 SRNs in each reservation for a given experiment, whereas another competitor might have selected a single reservation with 100 SRNs and a duration of 1-hour in preparation for an official scoring event. Both reservations would have had equivalent cost in tokens given the cost was a fixed cost per node-hour. The unit node-hour was the multiplication of the number of SRNs and the duration, in hours. On an average week, each competitor was able to access the Colosseum for 300 to 400 node-hours per week, or about 2% of the Colosseum's available node-hours. This allocation provided equal access across the competitors, while providing APL the ability to perform continuous upgrades on a rolling basis across a set of SRNs.

Alternatively, automated experiments did not require tokens to ensure equal access to Colosseum resources. Automated experiments were scheduled using a priority-based scheduler and, by design, gave equal access to all competitors. This algorithm is discussed in more detail later in this article.

The minimum time for a reservation was 20 minutes and included three stages: container allocation, experiment, and reservation cleanup. During container allocation, the competitor container was loaded on the reserved SRNs. Given the possible size of a competitor container (up to 20 GB) and the capacity of the management network, a maximum of 10 minutes was assumed for this action. During the experiment stage, the competitor interacted directly with the Colosseum services (in manual mode) or the Resource Manager orchestrated the services according to the experiment parameters (in automated mode). No assumptions on time were made for the experiment stage. In the final stage, the data were copied to the competitor's network storage and competitor containers were de-allocated. These actions returned the SRNs to a known-good state for the next competitor to use. Once again, given the size of these logs and the capacity of the management network, a maximum of 10 minutes was assumed for data transfer.

By design, competitors who submitted requests via the Colosseum website were valid since the drop-down boxes on the form only included valid responses (see

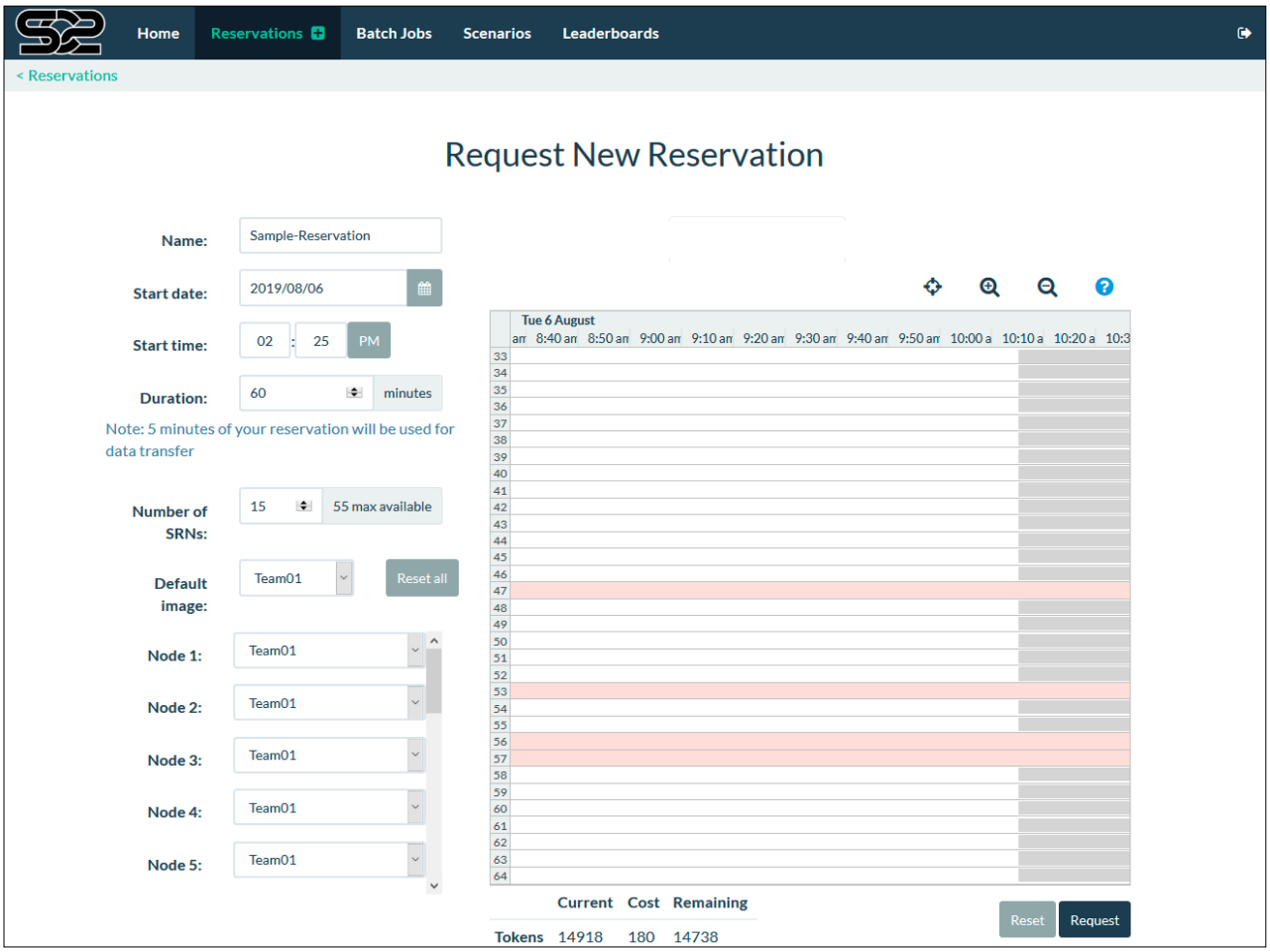


Figure 2. The SC2 reservation interface. Competitors exchanged tokens to reserve SRNs on the Colosseum. The shared calendar deconflicted reservation requests across competitors.

Figure 2). However, competitors were also able to submit reservation requests directly to the Resource Manager through online scripting. To ensure that the parameters were valid, the Resource Manager passed all requests to the request validator. Some examples of invalid reservation requests include a request for an experiment that was not listed on the Colosseum website, the competitor having insufficient tokens in their Colosseum account, or a request for a container that was not present on the Colosseum. Requests with invalid parameters were rejected and the competitor was notified via email of the error. Valid requests were passed to the scheduler within the Resource Manager. The scheduler placed the reservation on the Colosseum calendar.

AUTOMATED SCHEDULING ALGORITHM

For each experiment on the common experiment queue, the Resource Manager automatically scheduled the experiment on the Colosseum based on competitor priority and experiment size (measured in node-hours). To do this, the Resource Manager maintained a list of all registered competitors and sorted (prioritized) it based on the time of the competitor's last scheduled experiment (i.e., the competitor most recently scheduled was placed at the bottom of the list). Each competitor also maintained a local sorted list of experiments to be scheduled, with the highest-priority experiment on top. Competitors were responsible for maintaining the prioritization of their local lists. Using the prioritized competitor list and the individual competitor lists, the Resource Manager was able to schedule experiments on the Colosseum with the goal to maximizing the number of weekly experiments.

The scheduling algorithm in the Resource Manager was different in the first and second years of the competition. In the first year, experiments had a maximum size of 2.5 node-hours (10 SRNs with an experiment length of 15 minutes). Given 30 competitors and 128 SRNs (21,504 node-hours per week) in the Colosseum, a greedy scheduling algorithm was sufficient in the Resource Manager to automatically schedule experiments in the Colosseum with equal access. APL recognized that the greedy algorithm did not guarantee a globally optimal solution but selected the locally optimal choice at each stage in a reasonable amount of time. In other words, the Resource Manager guaranteed that an experiment would be scheduled for each scheduling attempt, but did not guarantee that the Colosseum schedule for the week was optimal.

The first-year algorithm (Algorithm 1) is shown in Figure 3 and was executed on regular fixed intervals. The input to the scheduling algorithm is the prioritized list of N-competitors, $C = \{C_1, C_2, \ldots, C_N\}$ and the N-competitor queues $Q = \{Q_1, Q_2, \ldots, Q_N\}$, where the head of each competitor queue holds that competitor's highest-priority experiment. For each scheduling interval, the algorithm attempted to schedule the first experiment from Q_1. If the algorithm succeeded, C_1 was placed at the bottom of C. If there were not enough SRNs to satisfy the first experiment, the algorithm attempted to schedule the next experiment from Q_1. The algorithm continued to schedule experiments from C_1's queue until it succeeded or tried all experiments in C_1's queue. If no experiments were scheduled, C_1 would remain at the top of C. The algorithm would then move on to C_2 while still within the interval. The scheduling algorithm would continue through C until an experiment was scheduled. Since the maximum size of an experiment was 2.5 node-hours, the algorithm was guaranteed to schedule an experiment per iteration.

Figure 4 illustrates the Colosseum calendar after four scheduling intervals (t_0, $t_0 + T$, $t_0 + 2T$, $t_0 + 3T$). At $t = t_0$, experiment e is selected from the top competitor's queue (purple competitor, denoted by an asterisk in Figure 4). The experiment requires i-SRNs (indexed [0] to [i-1]) and has a length of $(t_1 - t_0)$. At the next scheduling interval ($t_0 + T$), the next experiment is scheduled from the top

```
Algorithm taskSchedule(C,Q)
1.    Input:
2.       set C of competitors sorted in priority order
3.       set Q of tasks comprised of competitor queues
4.    Output: non-conflicting schedule

5.    on regular interval:
6.       until an experiment is scheduled or no experiments left to try:
7.          TC ← C[top]    // get top competitor from C
8.          QTC ← Q[TC]    // get top competitor's queue
9.          for all experiments, e in QTC:
10.            if size(e) is available in Colosseum:
11.               schedule e on Colosseum
12.               remove e from QTC
13.               move TC to bottom of C
14.               stop    // experiment has been scheduled
```

Figure 3. Greedy scheduling algorithm (Algorithm 1). In the first year of the competition, because experiments included only up to 10 SRNs, the Resource Manager used a greedy scheduling algorithm to schedule experiments in the common experiment queue.

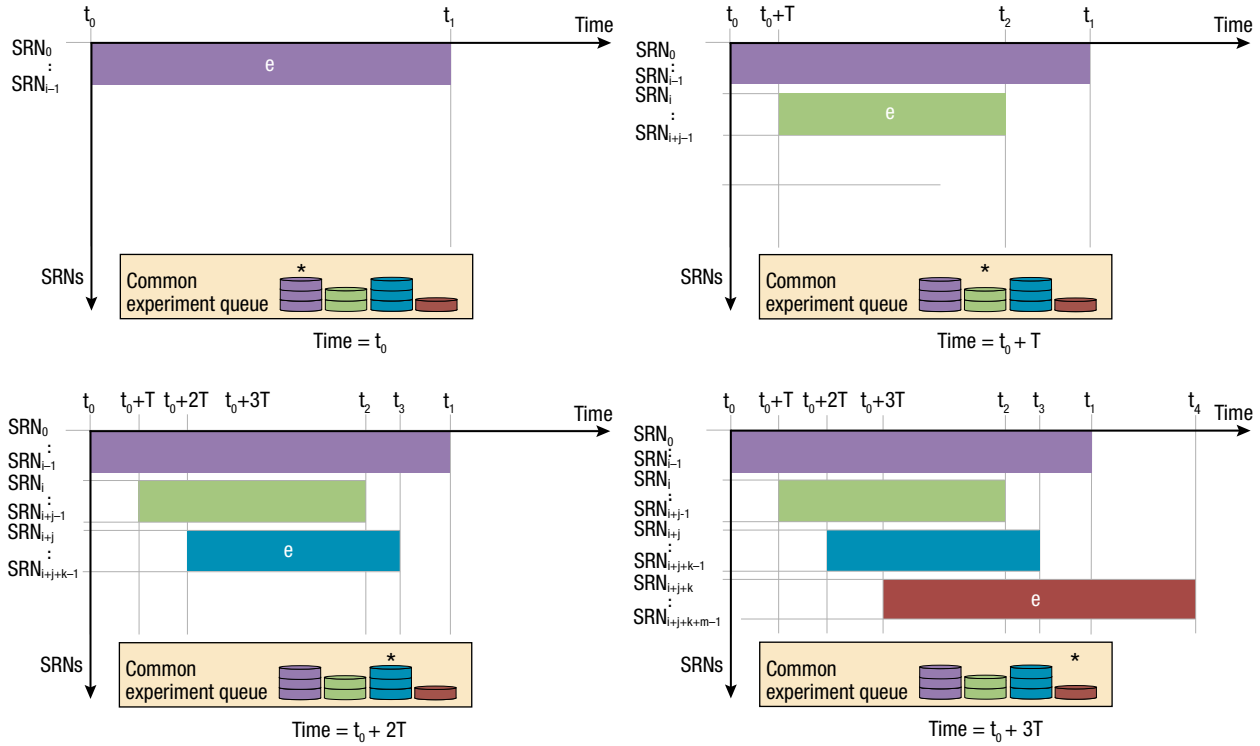


Figure 4. The greedy scheduling algorithm executed on fixed intervals and scheduled one experiment from the common experiment queue from the competitor with highest priority (denoted with an asterisk). The fixed intervals resulted in wasted Colosseum time after multiple iterations (four iterations are shown here).

competitor's queue (green competitor, denoted by an asterisk in Figure 4). The experiment requires j-SRNs and has a length of ($t_2 - t_0 = T$). Notice that no experiments are scheduled between t_0 and $t_0 + T$. This is an example of the greedy algorithm not achieving a globally optimal schedule (i.e., no wasted Colosseum time). The algorithm continues in this round-robin manner until all experiments are scheduled.

In the second (and third) year of the competition, the maximum size of experiments grew to 100 SRNs and an experiment length of 40 minutes, or 66.7 node-hours. In this case, the greedy approach was no longer ideal since it left large gaps in the Colosseum calendar and would favor experiments with small node-hour requirements over large experiments. This was not favorable to competitors, since the second and third years of competitions focused on proving that the competitor's spectrum access algorithms would work at scale. Therefore, APL modified the scheduling algorithm to prioritize large experiments over smaller ones. Additional modifications were made to the algorithm to improve Colosseum utilization by scheduling multiple experiments as close as possible within a given interval.

The updated algorithm (Algorithm 2) is shown in Figure 5. The input to the algorithm is identical to Algorithm 1, with N-competitors, $C = \{C_1, C_2, \ldots, C_N\}$ and the N-competitor queues $Q = \{Q_1, Q_2, \ldots, Q_N\}$, where the head of each competitor queue holds that competitor's highest-priority experiment. However, the scheduling interval t in Algorithm 2 is set based on the length of time required to execute the first experiment from C[top] (see line 9). In other words, the algorithm guarantees scheduling the top experiment from the highest-priority competitor (since the schedule is always empty in the proceeding interval), which ensures that the largest experiments are scheduled when placed on top of the competitor's queue. Once the scheduling interval was defined, the algorithm attempted to pack additional experiments into this interval as possible from the other competitor queues (without modifying their priority in C). Note that the primary experiment was scheduled in lines 11–13 in both Algorithm 1 and Algorithm 2. Lines 14–26 were added in Algorithm 2 to support packing additional experiments in the given time interval to improve Colosseum utilization. All competitor experiments were evaluated for packing in a round-robin manner, including the highest-priority competitor.

Figure 6 illustrates the Colosseum calendar for one scheduling interval, $[t_0, t_0 + t_1]$, where t_1 was set by the length of experiment eQTC (eQTC was selected at t_0 from the highest-priority competitor queue). Once eQTC was scheduled, the priority shifted to the next competitor. To improve Colosseum utilization, the algorithm attempted to schedule experiments from all competitors in a round-robin algorithm within the same

```
Algorithm taskSchedule(C,Q)
1.    Input:
2.       set C of competitors sorted in priority order
3.       set Q of tasks comprised of competitor queues
4.    Output: non-conflicting schedule

5.    on scheduled interval, t:
6.          TC ← C[top]    // get top competitor from C
7.          QTC ← Q[TC]    // get top competitor's queue
8.          eQTC ← QTC[top] //get top experiment from top competitor's queue

9        tFuture ← t + length(eQTC)
10.         Schedule next interval at tFuture
11.         Schedule eQTC on Colosseum
12.         Remove eQTC from QTC
13.         Move TC to bottom of C

14.         i ← 1
15.         while SRNs are available and experiments can be scheduled:
16.             nTC ← C[i] //get next competitor from C
17.             qTC ← Q[nTC] //get next competitor's queue
18.             for all experiments, e in qTC:
19.                 nFuture ← t + length(e)
20.                 if (size(e) is available) and (nFuture < tFuture)
21.                     schedule e on Colosseum
22.                     remove e from qTC
23.                     stop searching in qTC for experiments to schedule

24.             // move onto next competitor;
25.             // return to head of C after all competitors are checked
26.             increment i
```

Figure 5. Priority scheduling algorithm (Algorithm 2). In the second year of the competition, because experiments included up to 100 SRNs, the Resource Manager algorithm was enhanced to guarantee competitors' access to a large number of SRNs while also scheduling multiple experiments per interval.

time interval. After five iterations, no more experiments were scheduled within the interval. Note that in Algorithm 2 the unused Colosseum time in the schedule was dependent on the length of eQTC and the length of the other experiments in the common experiment queue versus the scheduling interval in Algorithm 1. In the second year, this change in the algorithm showed significant improvement and the improvement was greater as experiments started to have a standard size in the third year (50 SRNs, 15 minutes).

THE ORCHESTRATION SYSTEM

The Resource Manager coordinated the synchronization and execution of reservations (including experiments), regardless of scheduling modality. The Resource Manager had to allocate containers to SRNs that enabled (1) competitors to log in and start the experiment via the command line interface (in manual mode) and (2) boot scripts to execute from within the competitor container that triggered the start of an experiment

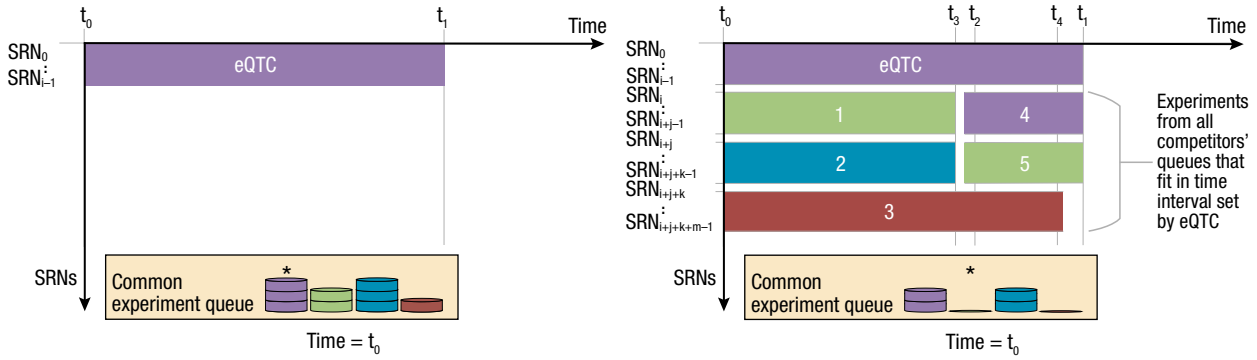


Figure 6. The improved scheduling algorithm executed on dynamic intervals set by the duration of the highest-priority experiment. Once the interval was set, additional experiments were selected from the common experiment queue to increase Colosseum utilization. This effort reduced the amount of wasted Colosseum time compared to Algorithm 1.

(in automated mode). In both cases, the allocation of containers and the start time of the experiment had to be uniform across the individual Colosseum services (traffic generation, RF emulation, GPS emulation). For example, if a reservation included 10 SRNs, the Resource Manager had to guarantee that all 10 SRN container images were loaded without error prior to the start of an experiment. Furthermore, the Resource Manager also had to guarantee that the wireless channel between the 10 SDRs was appropriately configured before the experiment started and that updates to the wireless channels between all 10 SRNs were synchronized throughout the experiment. This uniformity ensured that the experiment was consistent and repeatable across reservations, and thereby that the Colosseum (and research conclusion derived from data generated on the Colosseum) was verifiable.

The orchestrator was the entity within the Resource Manager that synchronized the Colosseum services and coordinated the stages of an experiment within a reservation. The orchestrator maintained bidirectional interfaces to the Traffic Generation System, the RF Emulation System, GPS emulation, and SRNs and, as such, was able to coordinate individual services (if requested by the competitor through the Command Line Interface [CLI]) and coordinate full experiments when in automated mode.

As previously described, the reservation included three stages: container allocation, experiment, and reservation cleanup. During the container allocation stage, the orchestrator allocated the competitor containers to the reserved SRNs. In addition to loading the container on the SRN, the SDR was programmed and the competitor network was configured. If any error occurred, the reservation was aborted and the SRNs were returned to the pool of available resources. Upon success, the SRN reported back to the orchestrator using the Radio API. If the SRN was reserved in manual mode, the competitor could log in to the SRN. If the SRN was reserved in automated mode, the orchestrator began the experiment.

At the start of an experiment, the orchestrator negotiated a start time across the three Colosseum services. This is important given the complexity of the individual services and the precision of emulation (e.g., the RF Emulation System had a 10-ns resolution for wireless channel filtering). If any error occurred in the individual system, it was reported to the orchestrator at this time. For example, if the Traffic Generation System could not provide sufficient resources for large data streams, the experiment would fail and the reservation would immediately move to the reservation cleanup stage. Upon success, the experiment was scheduled in the individual services and the individual services interacted with the SRNs as required for the duration of the experiment. Successful operations and nonfatal errors during the experiment were captured within each service and collected at the end of the experiment; fatal errors were captured by the orchestrator, resulting in an experiment abort and an immediate move to reservation cleanup.

At the end of an experiment, the orchestrator ended the individual services and began the reservation cleanup process on the SRNs. All the logs associated with the experiment from the individual services were moved to the competitor folder on the network-attached storage. Logs within the competitor container were also moved to the network-attached storage location to provide the competitor a complete record of the experiment. The orchestrator verified that the SRN and Colosseum services were returned to a known-good state before returning the SRNs to the pool of available resources.

CONCLUSION

APL implemented a custom Resource Manager to manage resources and organize experiments across the Colosseum for DARPA's SC2. A centralized approach ensured equal resource access and enabled automated orchestration of experiments. During the Colosseum's first year, the algorithm took a simple approach to scheduling, attempting to fill in the gaps in the Colosseum's schedule. During the second and third years, as the experiments included more SRNs and automated scheduling became the default mode of operation, APL improved the scheduling algorithm to maintain competitors' equal access to Colosseum resources. As a result, each competitor had equal access to Colosseum resources in the three years of competition, and APL was able to increase the speed of research for next-generation wireless communication systems.

ACKNOWLEDGMENTS: We thank Paul Tilghman (DARPA SC2 program manager) and Craig Pomeroy and Kevin Barone (Systems Engineering and Technical Assistance at DARPA) for their invaluable collaboration and support. We also thank the many APL SC2 contributors, whose names are listed on the inside back cover of this issue of the *Digest*. This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the US government.

REFERENCE

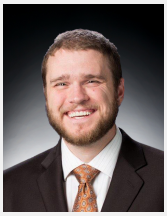
[1]"SC2, Spectrum Collaboration Challenge." DARPA. https://www.spectrumcollaborationchallenge.com/ (accessed Aug. 28, 2018).

Jason W. Mok, Asymmetric Operations Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Jason W. Mok is a software engineer at APL with an MS in computer science from the University of Texas at Dallas. Jason served as software lead for the DARPA Spectrum Collaboration Challenge (SC2) Colosseum in phases 2 and 3, making contributions to the website, the Resource Manager, the SRN controller, and the RF Emulation System. Jason has contributed to multiple efforts to progress cybersecurity automation and the intelligent control plane and has an interest in DevOps and version control. His email address is jason.mok@jhuapl.edu.

Alexander L. Hom, Asymmetric Operations Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Alexander L. Hom is a software engineer at APL. He received a BS in software engineering from Rochester Institute of Technology in 2013. He received an MS in computer science from Johns Hopkins University in 2015. He contributed to the design and implementation of the Resource Manager for DARPA's Spectrum Collaboration Challenge (SC2). His email address is alexander.hom@jhuapl.edu.



Jason J. Uher, Asymmetric Operations Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Jason J. Uher is a member of the Senior Professional Staff and a chief scientist in APL's Asymmetric Operations Sector. He earned a BS in computer and electrical engineering from the University of Nebraska-Omaha, an MS in electrical and computer engineering from the Georgia Institute of Technology, and a PhD in engineering from the University of Nebraska-Lincoln. In the chief scientist role, Jason leads science and technology (S&T) efforts in the Wireless Cyber Capabilities Group by promoting staff involvement in research projects and providing direction for S&T planning and investments. For DARPA's SC2, he led the competitor user experience software team, which was tasked with providing all automation and orchestration of competitor software and results reporting. His email address is jason.uher@jhuapl.edu.

David M. Coleman, Asymmetric Operations Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

David M. Coleman is research area lead for spectrum operations and supervisor of the Communication Systems Concepts Section in APL's Asymmetric Operations Sector. He earned a BS in computer engineering from Elizabethtown College, an MS in computer engineering from the University of Arkansas, and a PhD in electrical engineering from University of Maryland, College Park. For the DARPA SC2 effort, Dr. Coleman led the software development for the Colosseum's RF Emulation System as well as the Interface Control Document definition between this subsystem and the Colosseum. Dr. Coleman has presented at many conferences and has been published in journals and proceedings. His email address is david.coleman@jhuapl.edu.