



Jet Propulsion Laboratory
California Institute of Technology

Enabling Limited Resource-Bounded Disjunction in Scheduling

Jagriti Agrawal, Wayne Chi, Steve Chien, Gregg Rabideau, Stephen Kuhn, and Dan Gaines

Background

- The Mars 2020 Scheduler is a (Rabideau and Benowitz 2017)
 - single-shot, non-backtracking scheduler that
 - schedules in *priority first order* and
 - never removes or moves an activity after it is placed during a single scheduler run.
 - Activities are not preempted
 - It does not search except for
 - Valid intervals calculations
 - sleep and preheat scheduling

Challenge

- Working with a very computationally limited scheduler that does not backtrack
 - Conservative estimate of 60 seconds to run
- How can we get more out of the scheduler and improve the final schedule given that the scheduler highly computationally limited?

Goal

- Primary goal is to schedule all **mandatory activities**- high priority activities that must be scheduled
- Would also like to schedule certain preferred activities as long as doing so does not prevent any mandatory activities from being scheduled
 - Grouping of activities into *switch groups*
- **Switch Group**- set of activities where exactly one of the activities in the set **must** be scheduled
 - Activities within a switch group are called *switch cases*
 - Generally, switch cases differ only by the amount of resources (time, energy, data volume) they consume
 - Goal is to schedule more resource-consuming switch case without dropping another mandatory activity from the schedule
 - Challenge- scheduler is non-backtracking, no search, uses a greedy algorithm to place activities
 - Example:

$$\text{SwitchGroup} = \left\{ \begin{array}{l} \text{Activity A} = 100 \text{ sec} \\ \text{Activity B} = 200 \text{ sec} \\ \text{Activity C} = 400 \text{ sec} \end{array} \right.$$

- Must schedule one of A, B, or C
- Would like to schedule C if does not force another mandatory out of schedule

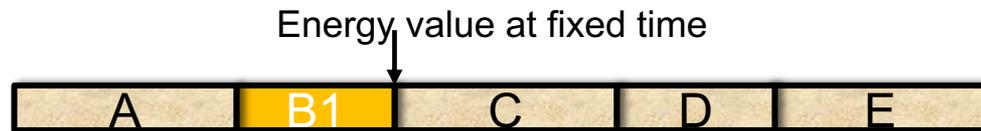
Various Methods to Schedule Switch Groups

- During execution, activities may end earlier and free up resources
 - How can we use extra resources to schedule more preferred switch case without dropping another mandatory activity?
- Methods:
 - **Guard Methods:** When scheduling switch cases, reserve enough sensitive resources (time, energy, data volume) to schedule remaining mandatory activities
 - For switch groups this means that resources will be reserved for the least resource consuming (minimum) activity
 - A) Fixed Point Guard
 - B) Sol Wide Guard (Sol- Martian Day)
 - All computations done offline
- **MSI (Multiple Scheduler Invocation)**
 - During execution, trigger a special process (MSI) that reinvokes the scheduler multiple times, once for each level of the switch group
 - Only considers the next switch case (in terms of time)
 - Emulates backtracking at a very limited scale

Two Types of Guard Methods

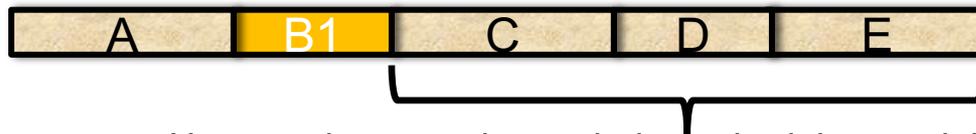
- Both methods guard for unscheduled activities after (scheduling order) the switch case regardless of when they will be executed, e.g. they are scheduled in a later loop of the scheduler but temporally before the switch group
- Ex) Switch Group: $B1 < B2 < B3$, Scheduling Order: A, B1, C, D, E (not necessarily same as order of temporal placement)

- **Fixed Point**



- **Sol Wide**

- Guards for energy by keeping track of the energy balance in the entire sol versus at a fixed point



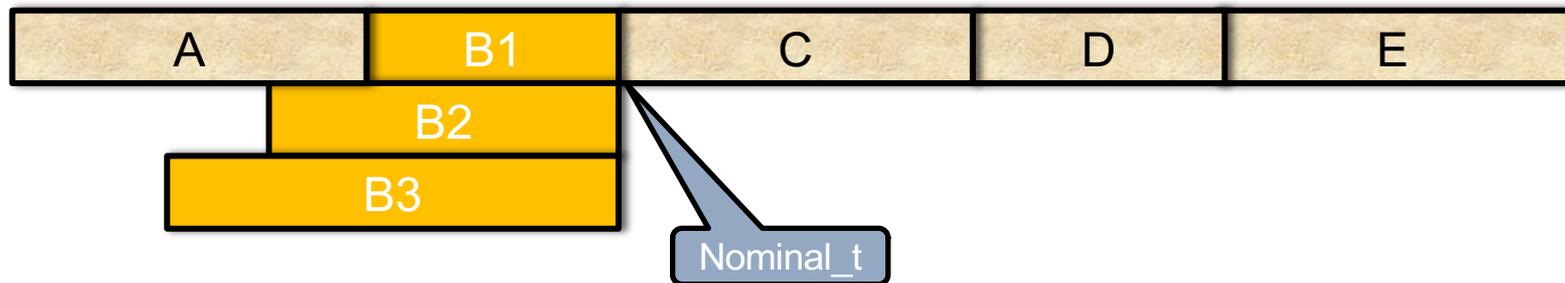
- The way these two methods guard for time is the same, they differ in how they guard for energy
- All computations to find values for guard constraints are done offline, and they are applied while scheduling during execution

Assumptions

- Nominal Plan = resulting plan after scheduling with **only the least resource consuming (minimal)** switch case
 - All activities have original, conservative duration
 - Should schedule all mandatory activities for guards to be viable
- Fixed Point- heavily based on assumption that activities will execute in same order as they were scheduled in Nominal Plan
- Sol Wide- Often, it is ok if order of execution order is different from scheduling order in nominal plan since we look at energy balance throughout the whole sol, not at a fixed point

Guarding Against Time (Same for both Guard Methods)- Option 1

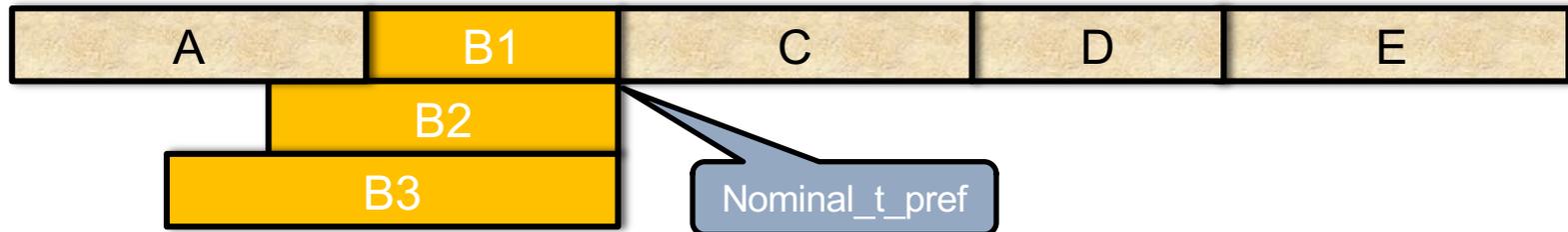
- **Nominal Plan** = plan after scheduling with only nominal (minimal) switch cases and nominal durations, should schedule all mandatory activities
- **Nominal_t** = time at which minimal switch case is scheduled to end execution in nominal plan



- Set execution time constraints (latest allowed start time) so that B2 or B3 must end by **Nominal_t**

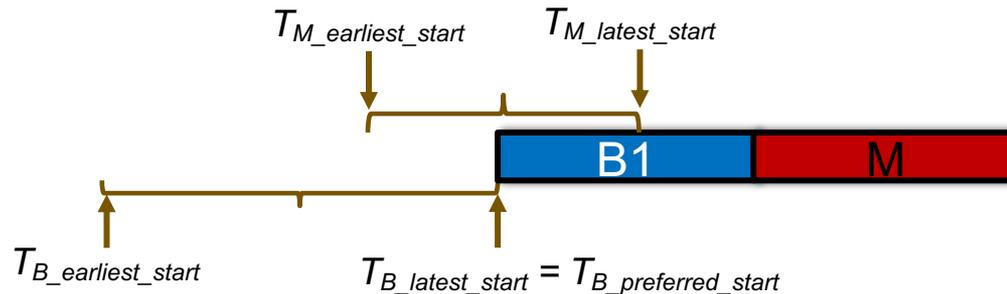
Guarding Against Time With Preferred Time-Option 2

- Nominal Plan = plan after scheduling with only nominal (minimal) switch cases, should schedule all mandatory activities
- Before generating the nominal schedule, set preferred time of nominal switch case to its latest start time given by execution time constraint
 - → Scheduler will try to schedule nominal switch case as late as possible in the nominal schedule
 - → Longer switch cases may have later latest start times, allowing for wider execution window
- Nominal_t_pref = time at which minimal switch case is scheduled to end execution in nominal plan using the set preferred time



- Set execution time constraints so that B2 or B3 must end by Nominal_t_pref
- Note: We DO NOT set a preferred time for B2 or B3, the scheduler will still try to schedule them as early as possible

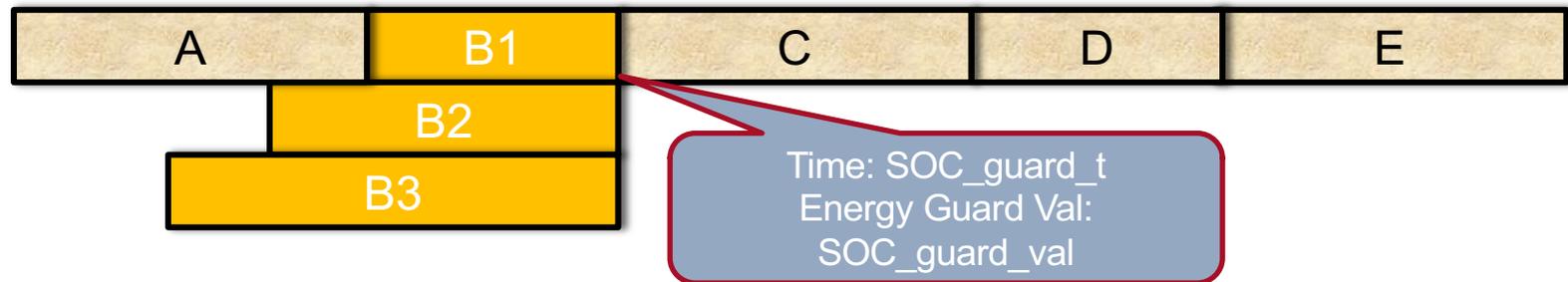
Guarding Against Time with Preferred Time-Caveat



- Possible for mandatories to be dropped when we set preferred time of minimum switch case to be latest start time
- M and B1 cannot overlap due to resource conflicts
- If nominal switch case is placed as late as possible, it could use up time from another mandatory activity with a tight execution window

Fixed Point Guard Minimum SOC

- Minimum SOC- State of charge cannot go below this value at any point during scheduling or execution
- Nominal Plan = plan after scheduling with only nominal (minimal) switch cases, should schedule all mandatory activities
- SOC_guard_t = time at which minimal switch case is scheduled to end execution in nominal plan
- SOC_guard_val = State of charge value at time SOC_guard_t



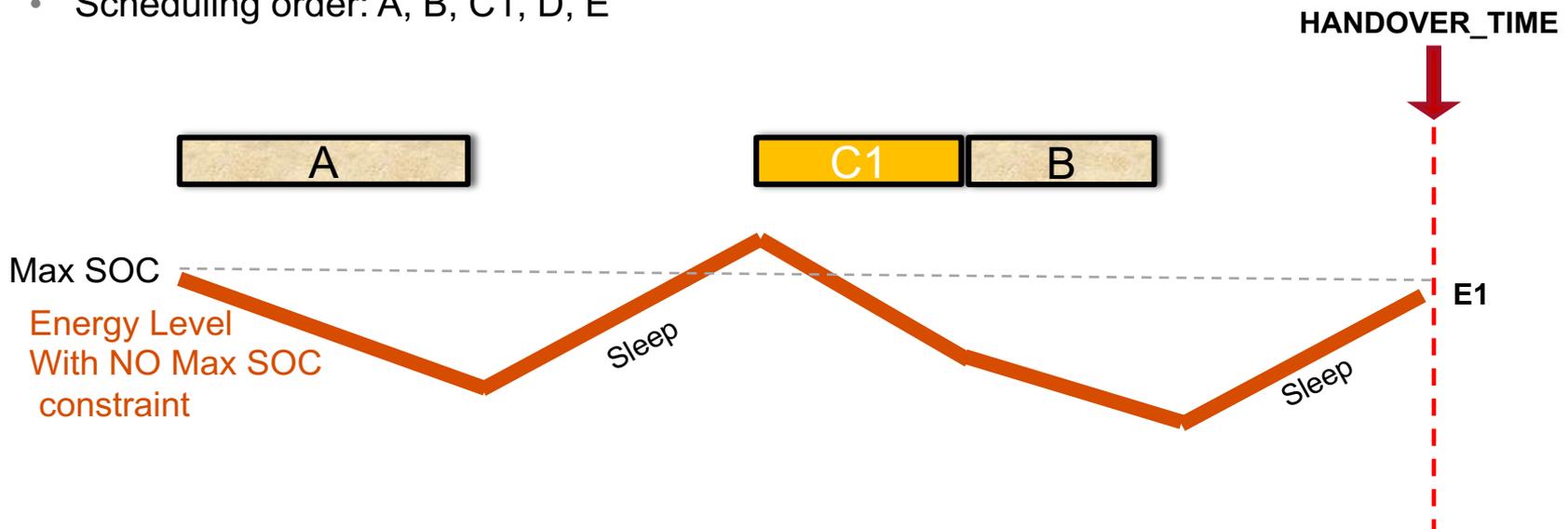
- Energy value cannot go below SOC_guard_val at time SOC_guard_t

Fixed Point Guard Handover SOC

- Handover time- in effect, time at which next schedule starts
- Handover SOC- state of charge cannot go below this value at handover time
- Nominal Plan = plan after scheduling with only nominal (minimal) switch cases, original durations, should schedule all mandatory activities
- Find how much extra energy is left at handover time after scheduling all activities in Nominal Plan
 - $\text{Energy leftover} = \text{Energy}(\text{Handover_time}) - \text{MINIMUM_HANDOVER_SOC}$
- Make sure extra energy needed to schedule longer switch case does not exceed energy left over
 - If it does, do not schedule switch case

Sol Wide Guard- Handover SOC Guard

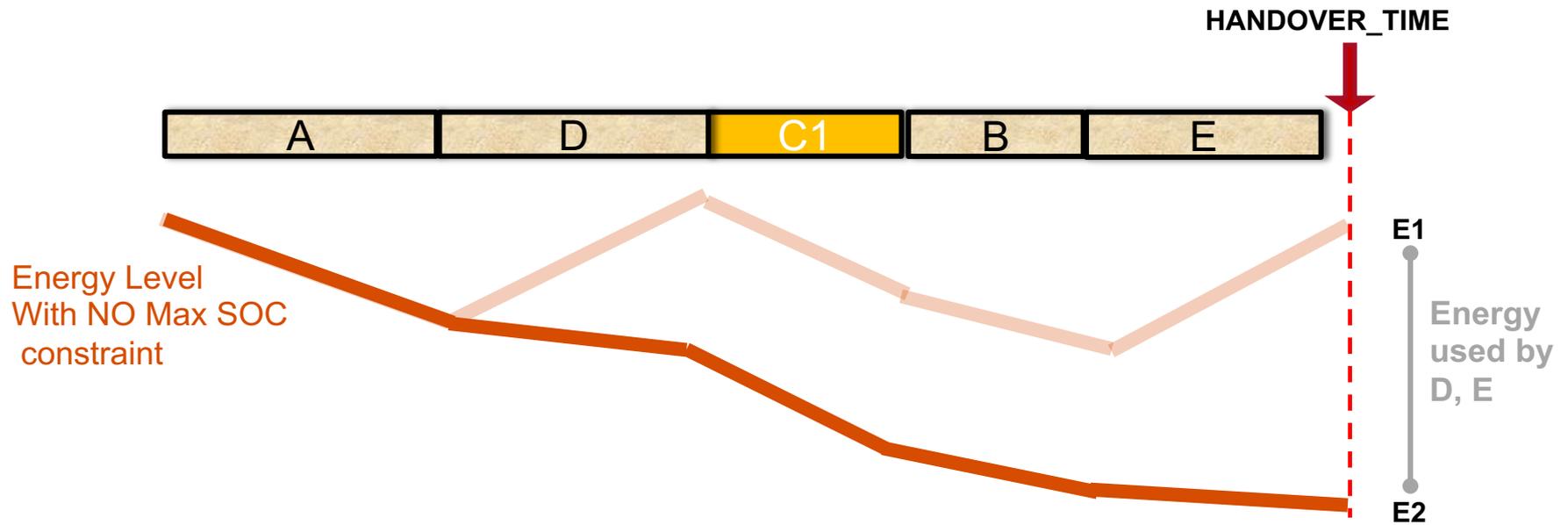
- How much energy is needed to schedule remaining mandatory activities?
- Maximum SOC- Energy cannot exceed this value at any point during plan execution or scheduling (rover may be kept awake to prevent it from exceeding this value)
 - Having this constraint may produce inaccurate result since any energy exceeding Max SOC would not be taken into account
- Remove Max SOC constraint while computing guard with nominal schedule offline to accurately keep track of energy balance
- Scheduling order: A, B, C1, D, E



E1 is energy level of nominal schedule with no Max Soc constraint after all activities up to and including the nominal switch case (A, B, C1) have been scheduled

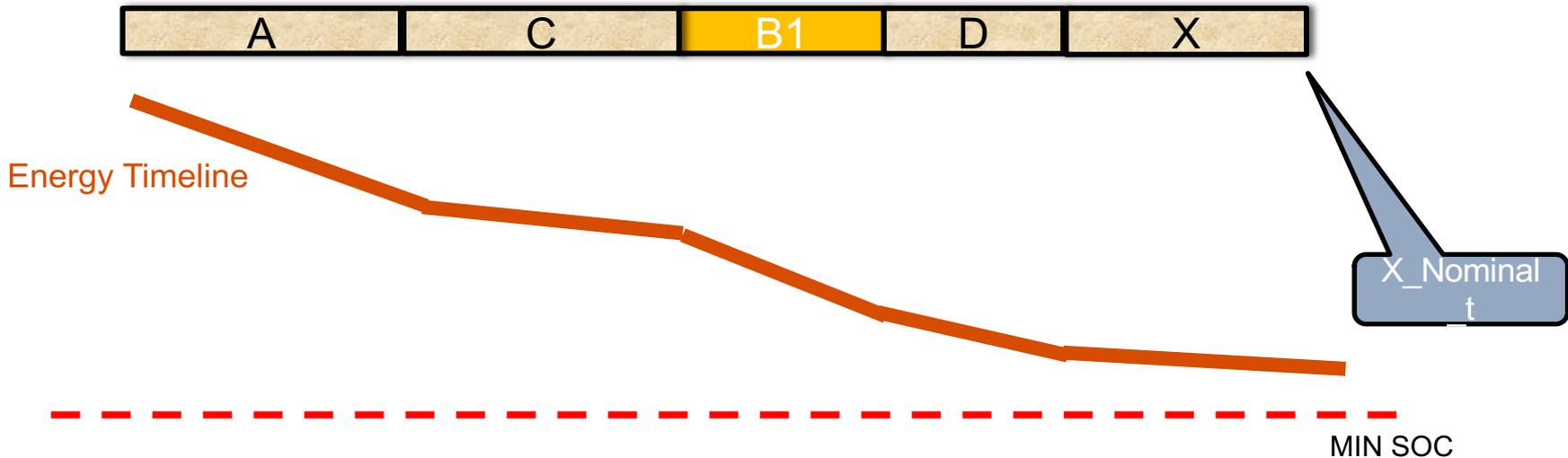
Sol Wide Guard- Handover SOC Guard

- Guarding against handover SOC
- E2 is energy level in nominal schedule with no Max Soc constraint after all activities in the plan have been scheduled
- Energy needed for activities after Bi (in terms of scheduling order, not time) = E1 - E2
- Guard value = MIN_HANDOVER_SOC + (E1 - E2)
- Scheduling order: A, B, C1, D, E



Guarding against Min SOC Sol Wide Guard

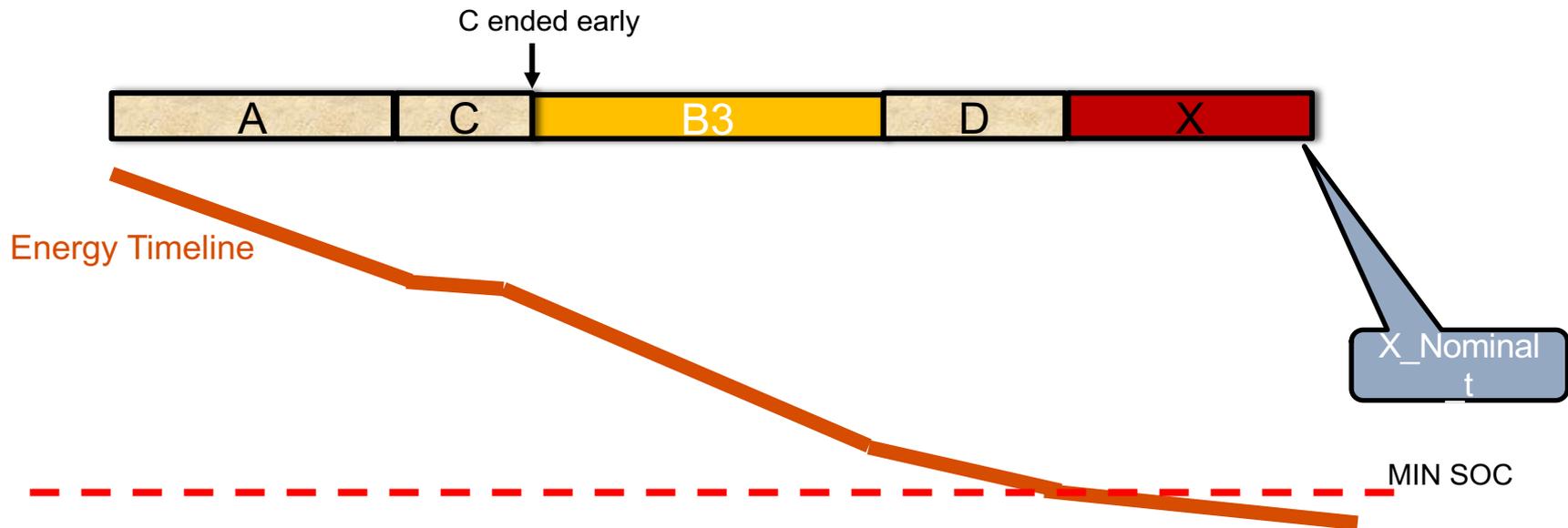
- Nominal Plan = plan after scheduling with only nominal (minimal) switch cases, should schedule all mandatory activities
- X = mandatory activity
- X_nominal_t = time at which activity X is scheduled to end execution in Nominal Plan



- Run Monte Carlo of execution with input plan with switch cases and guards
 - If X was NOT SCHEDULED and longer switch case was scheduled before X_nominal_t, then increase energy guard value for longer switch case

Monte Carlo for Min SOC Sol Wide Guard

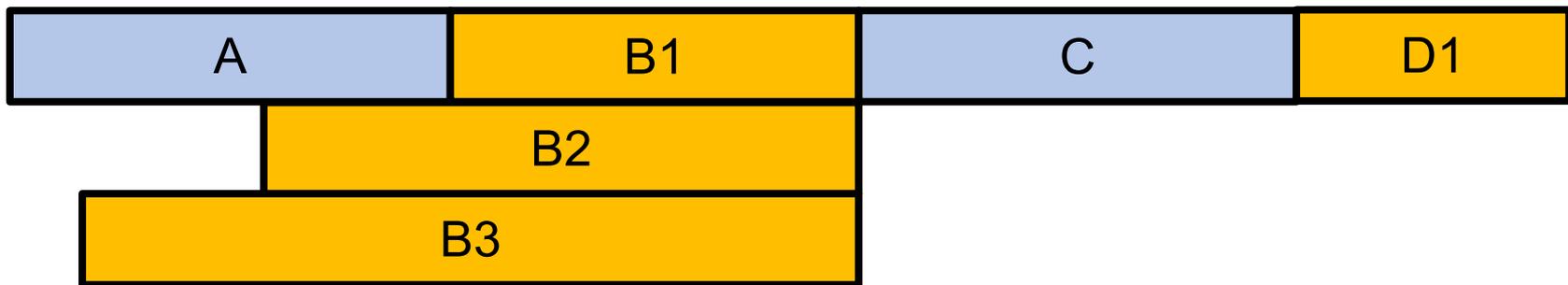
Execution run with switch cases and guards:



- X was not scheduled because energy dipped below MINIMIM SOC
 - Since B3 was scheduled before X_nominal_t, then increase energy guard value for longer switch case

Multiple Scheduler Invocation (MSI) Switch Groups

- 1) Generate schedule with only the nominal (lowest) switch group activity
- 2) During execution, trigger a special process (MSI) that reinvokes the scheduler multiple times, once for each level of the next switch group, to emulate backtracking.
 - Each switch group and its different levels are only attempted once.
 - When do we start reinvoking the scheduler?



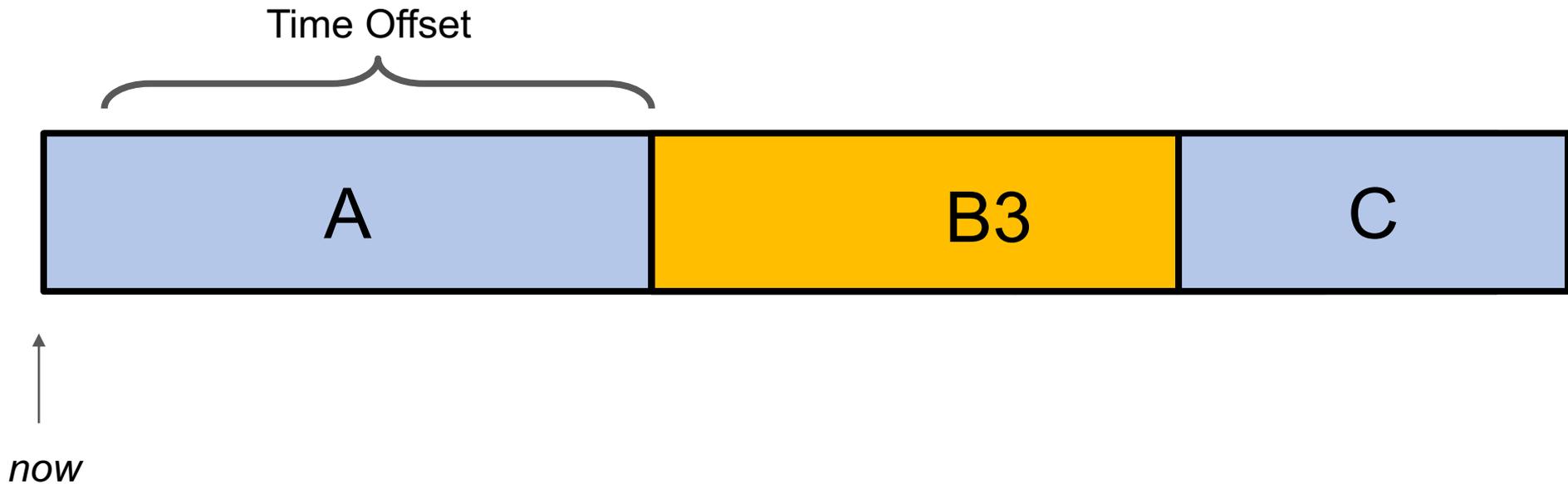
Create a nominal schedule with only B1. During MSI, attempt B3 then B2 once each. If both fail, possibly attempt B1 again.

When to Trigger MSI?

- Time Offset:
 - Begin Multiple Scheduler Invocations when $\text{now} = \text{switch activity scheduled start} - X$
- Switch Ready:
 - Begin Multiple Scheduler Invocations when
 - A switch activity is the next (scheduled start time) in the schedule.
 - An activity finished executing

Time Offset Trigger

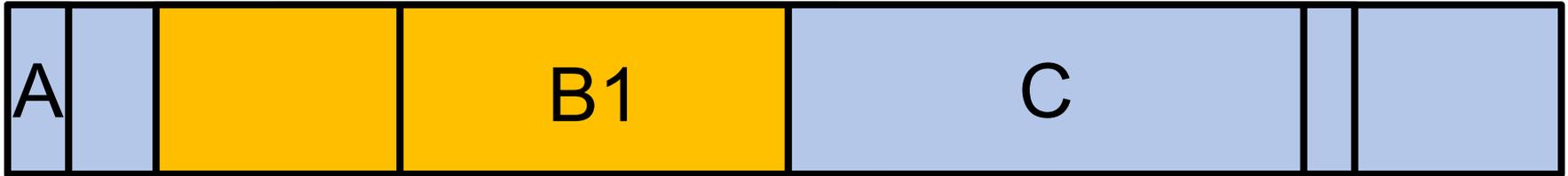
If B2 and C are both scheduled (and no handover constraints are violated), the process is finished.
If not, then B1 is attempted.



Switch Ready Trigger (SWRE)

If an elevated switch group is able to be scheduled without bumping out any other mandatory activities or violating handover constraints, then the process completes.

If an elevated switch group isn't able to be scheduled without violating any of the above, attempt the next lower switch group activity.



↑
now

Spacing Between MSI Invocations

- Choose to reschedule as soon as possible after the most recent MSI invocation
 - Risks overconsumption of CPU if scheduler is invoked too frequently
 - May need to use *throttling*- imposes a minimum time delay between invocations
- Other option- reschedule at evenly split fixed cadence (future work)

How do we space out the
rescheduling attempts during
MSI?

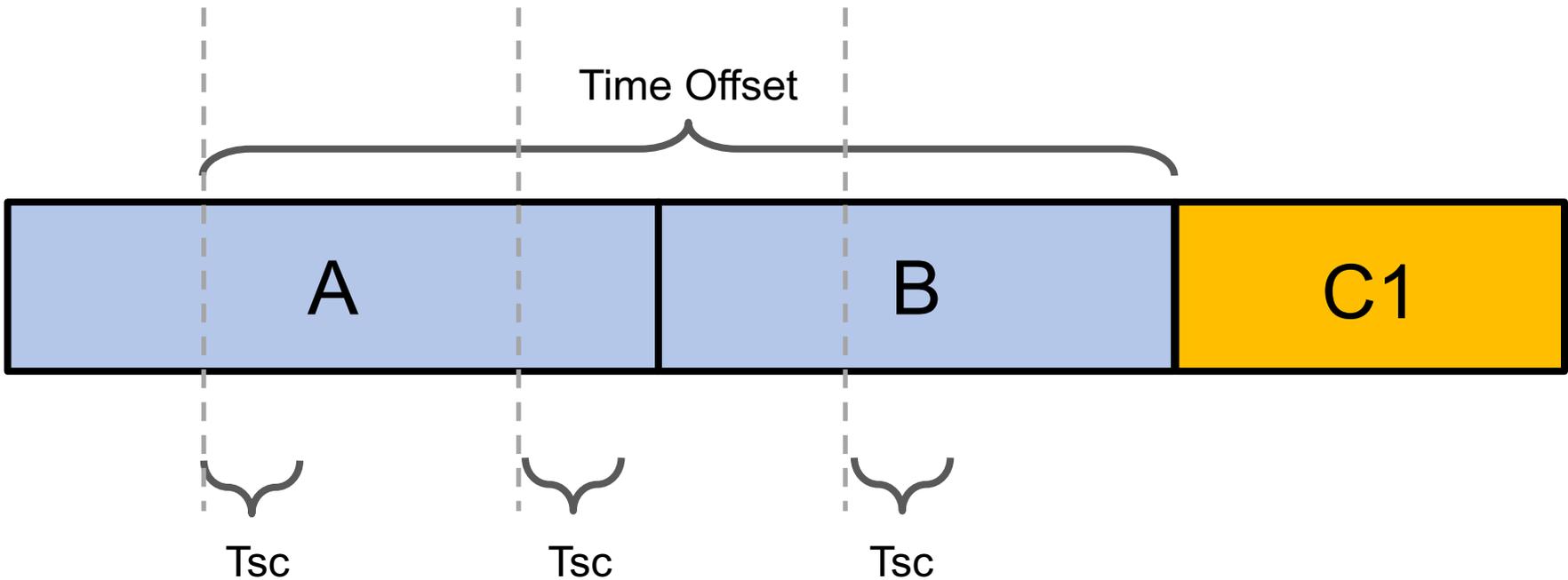
MSI Invocation Spacing

Time Offset



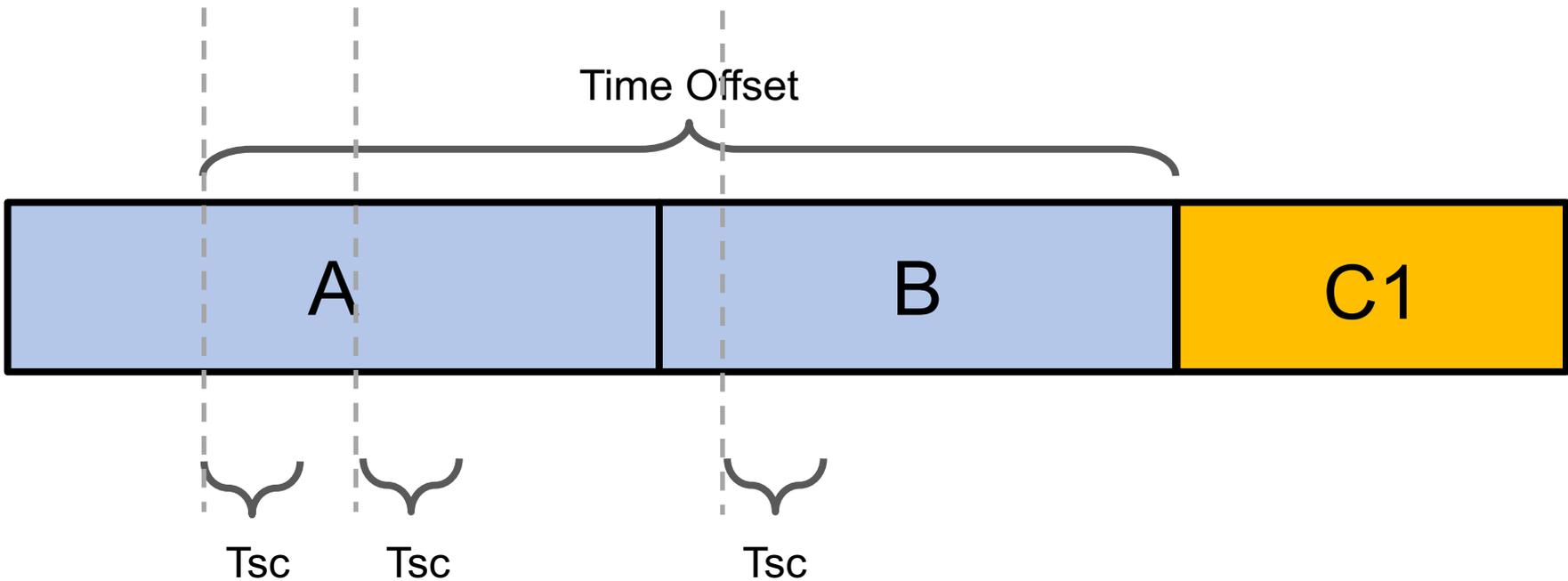
Start MSI and begin rescheduling

MSI Invocation Spacing (Fixed Cadence)



Reschedule at an evenly split Fixed Cadence. If the offset is large enough, no throttling will be required.

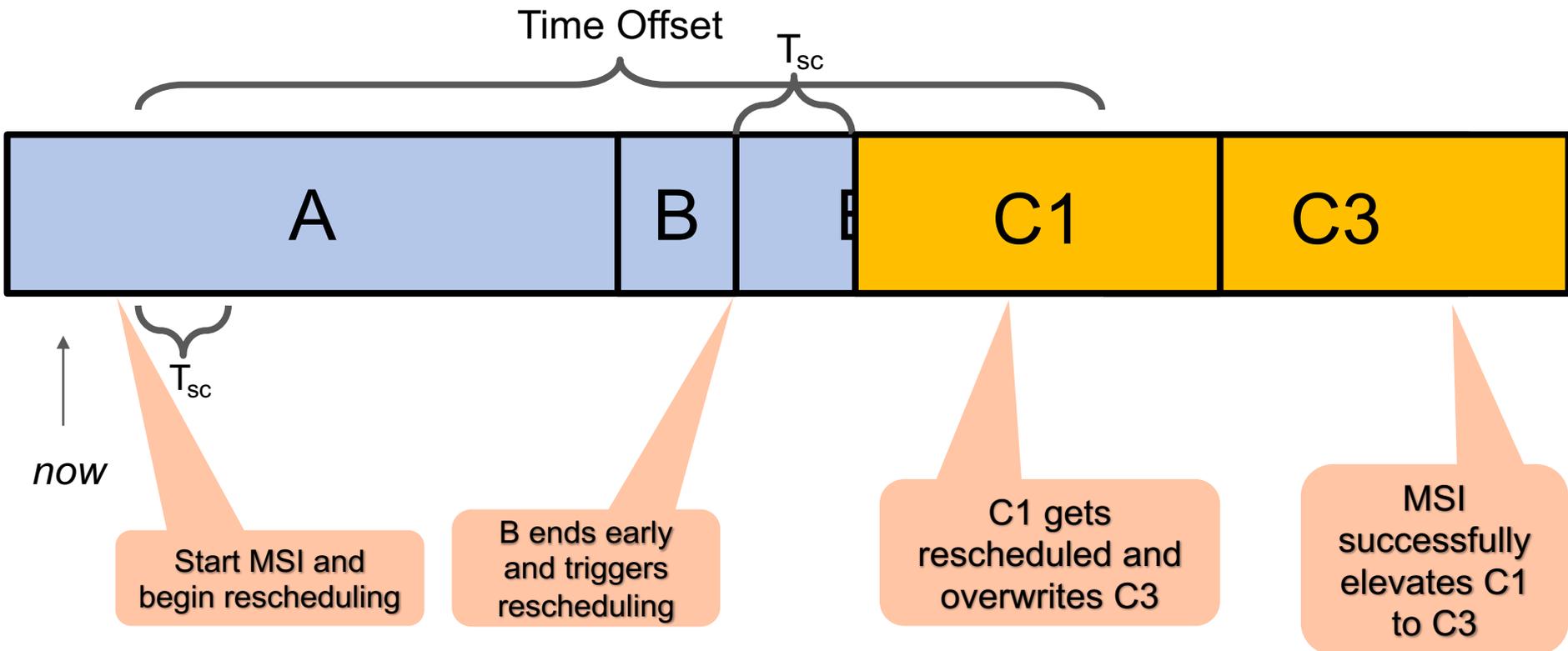
MSI Invocation Spacing (Earliest Possible w/ Throttling)



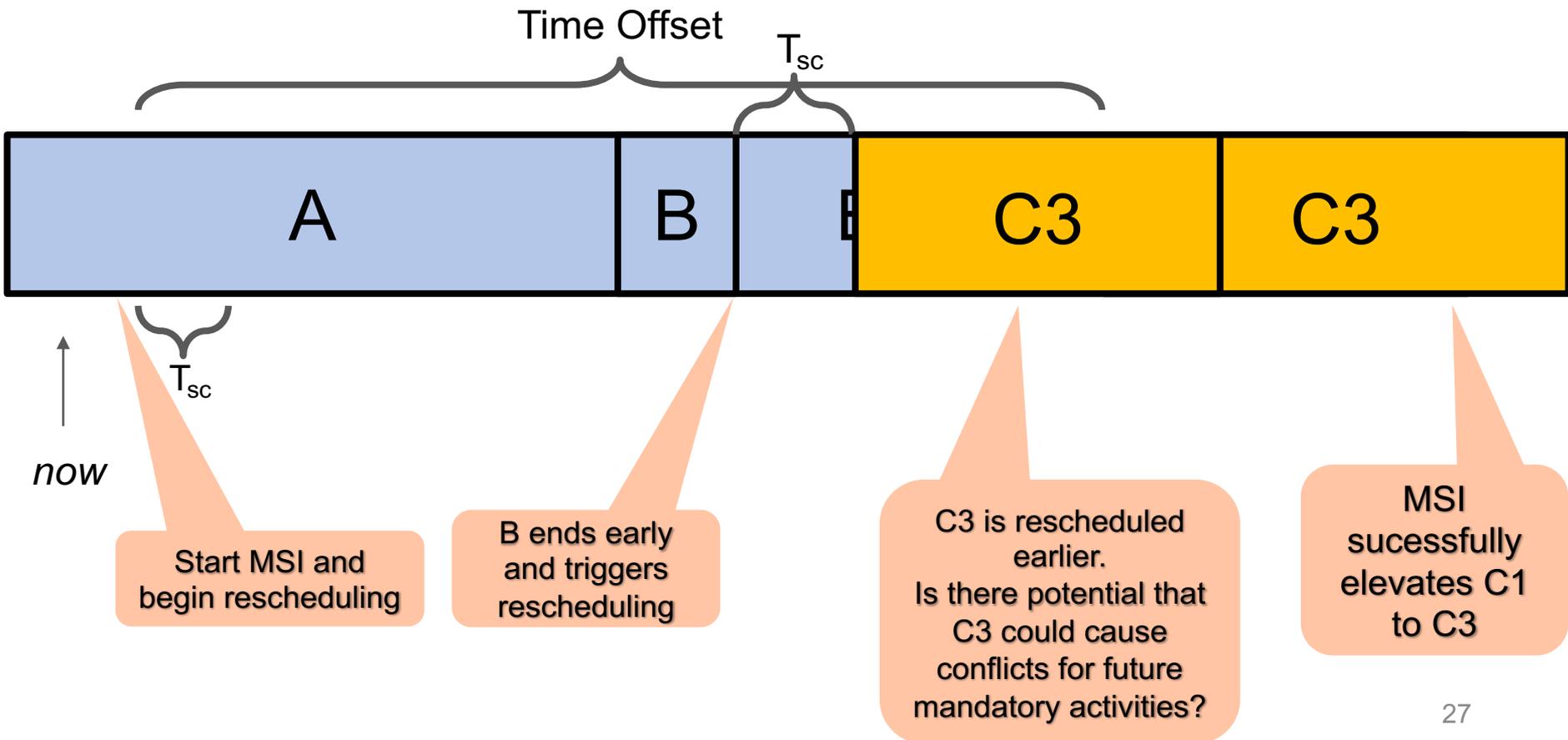
Reschedule as early as possible and use throttling to prevent overconsumption of the CPU.

Problem- Rescheduling after MSI but before SG is committed

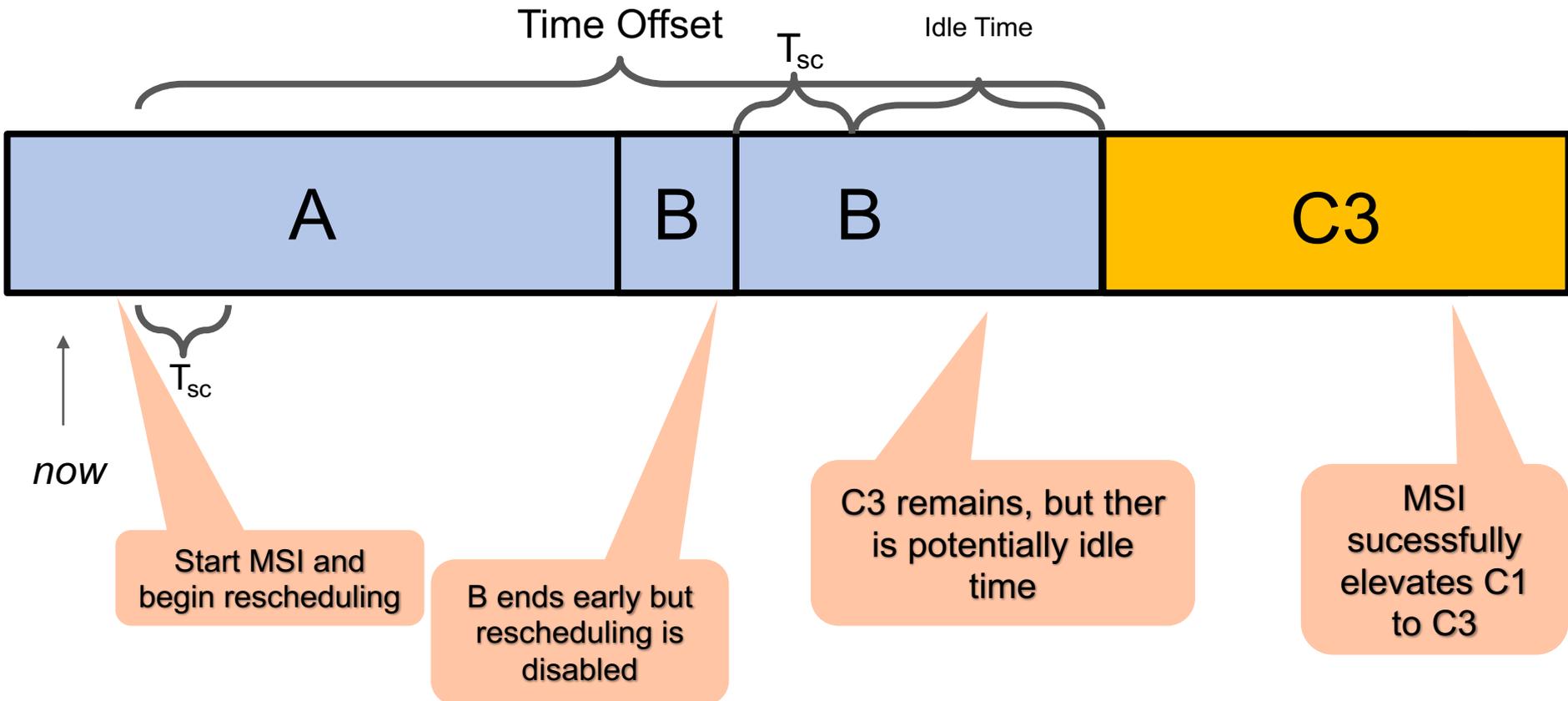
After MSI, there may be events that trigger rescheduling before the switch case is committed- scheduler must know which level switch case to consider



Solution 1)- Reschedule with Elevated Switch Group

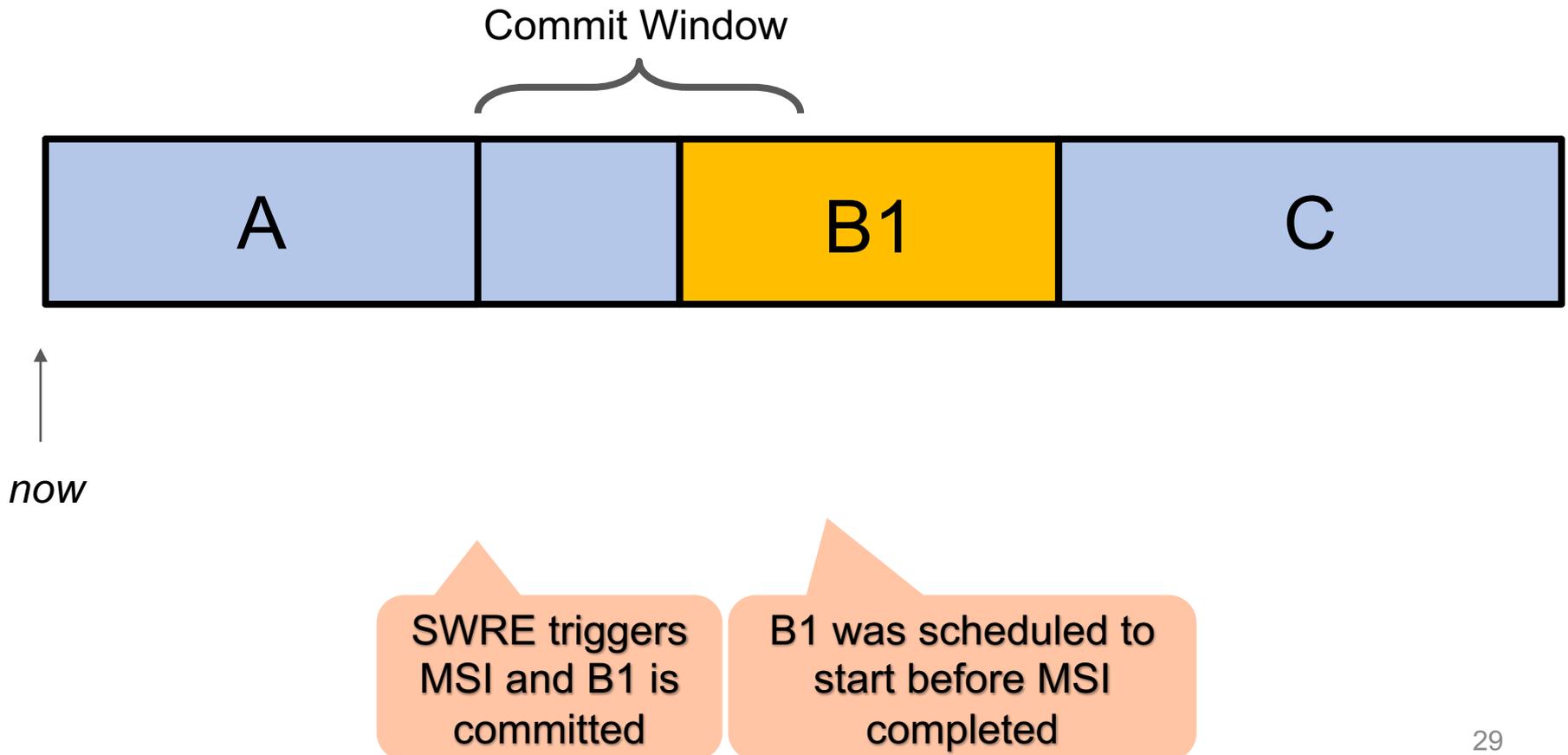


Solution 2) Disable Rescheduling until SG is committed



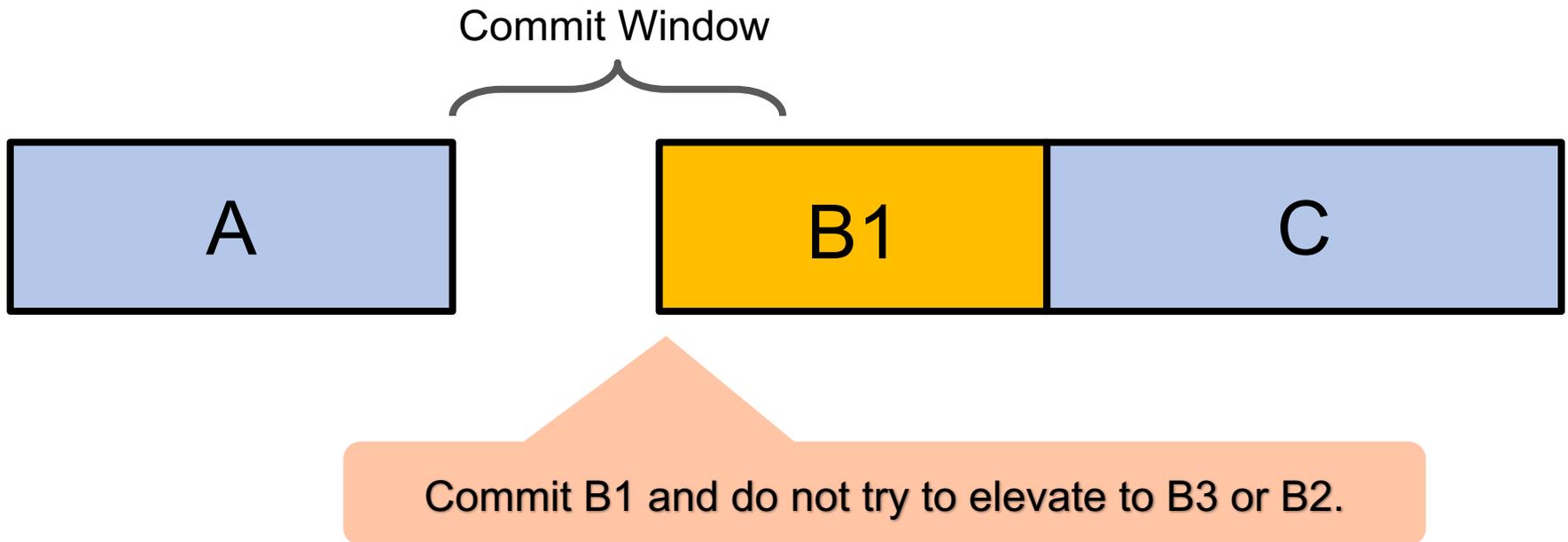
How to deal with a committed switch group activity?

In some situations, the next activity may become committed before (or during) multiple scheduler invocations.



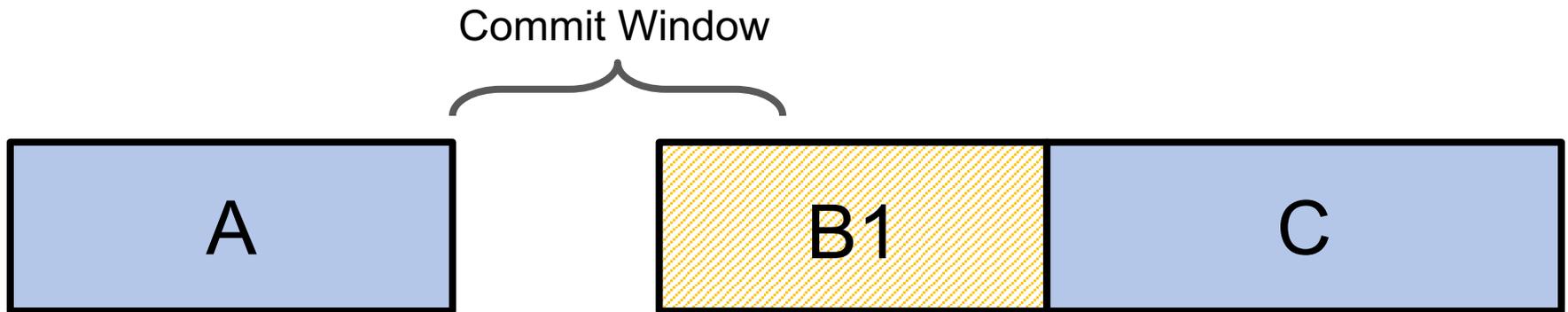
How to deal with a committed switch group activity? (Commit and Give Up)

In some situations, the next activity may become committed before (or during) multiple scheduler invocations.



How to deal with a committed switch group activity? (Veto)

In some situations, the next activity may become committed before (or during) multiple scheduler invocations.



Veto B1 and attempt B3 and B2. When an activity is *vetoed*, it is removed from the current schedule.

Empirical Results- Inputs

- Use Sol Types
 - **Sol Types**- currently best available data on expected M2020 rover operations
 - Each sol type contains 20-40 activities
 - Each sol type has a different objective (e.g. driving, more drilling, etc.)
 - 8 sol types, 10 variants per sol type
 - Each variant has one switch group only, with 3 switch cases

$$\text{SwitchGroup} = \left\{ \begin{array}{l} \text{Activity A} = x \text{ sec} \\ \text{Activity B} = 2x \text{ sec} \\ \text{Activity C} = 4x \text{ sec} \end{array} \right.$$

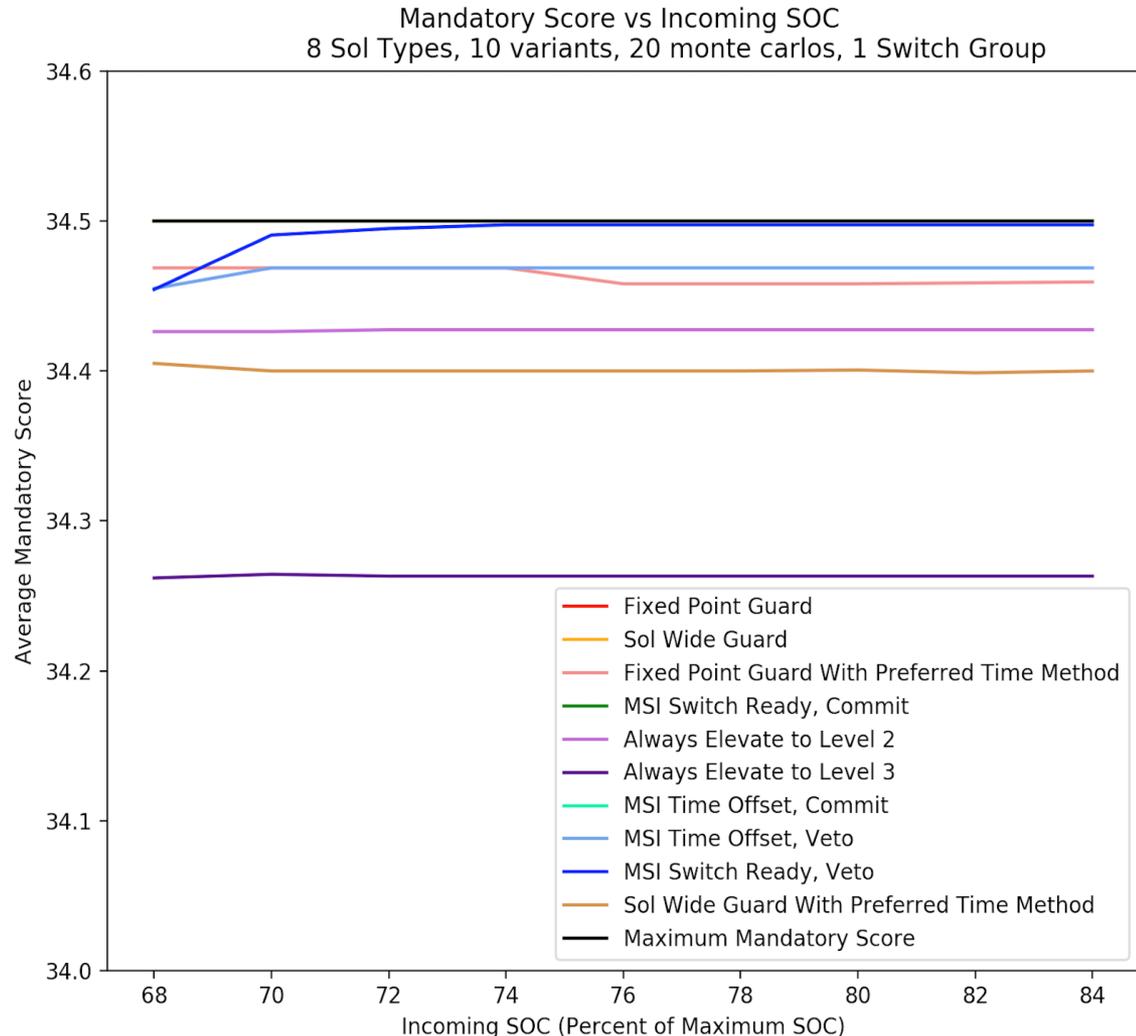
- 20 runs of simulation of execution for each variant
 - Use Mars 2020 surrogate scheduler- an implementation of same algorithm as Mars 2020 onboard scheduler but intended for a Linux workstation environment
 - Activities end on average 10% early
- Want to determine which methods result 1) scheduling all mandatory activities, 2) highest switch group score

Scoring

- Mandatory Activity Score
 - Each mandatory activity that is scheduled, including whichever switch case is scheduled contributes 1 point to the mandatory score
- Switch Group Score
 - A successfully scheduled switch case that is 4x as long as nominal contributes 1 point to switch group score
 - A successfully scheduled switch case that is 2x as long as nominal contributes 0.5 point to switch group score
 - If only the nominal switch case is able to be scheduled, it does not contribute to switch case score
- Scheduling a mandatory activity is of much higher importance than scheduling any number of switch cases

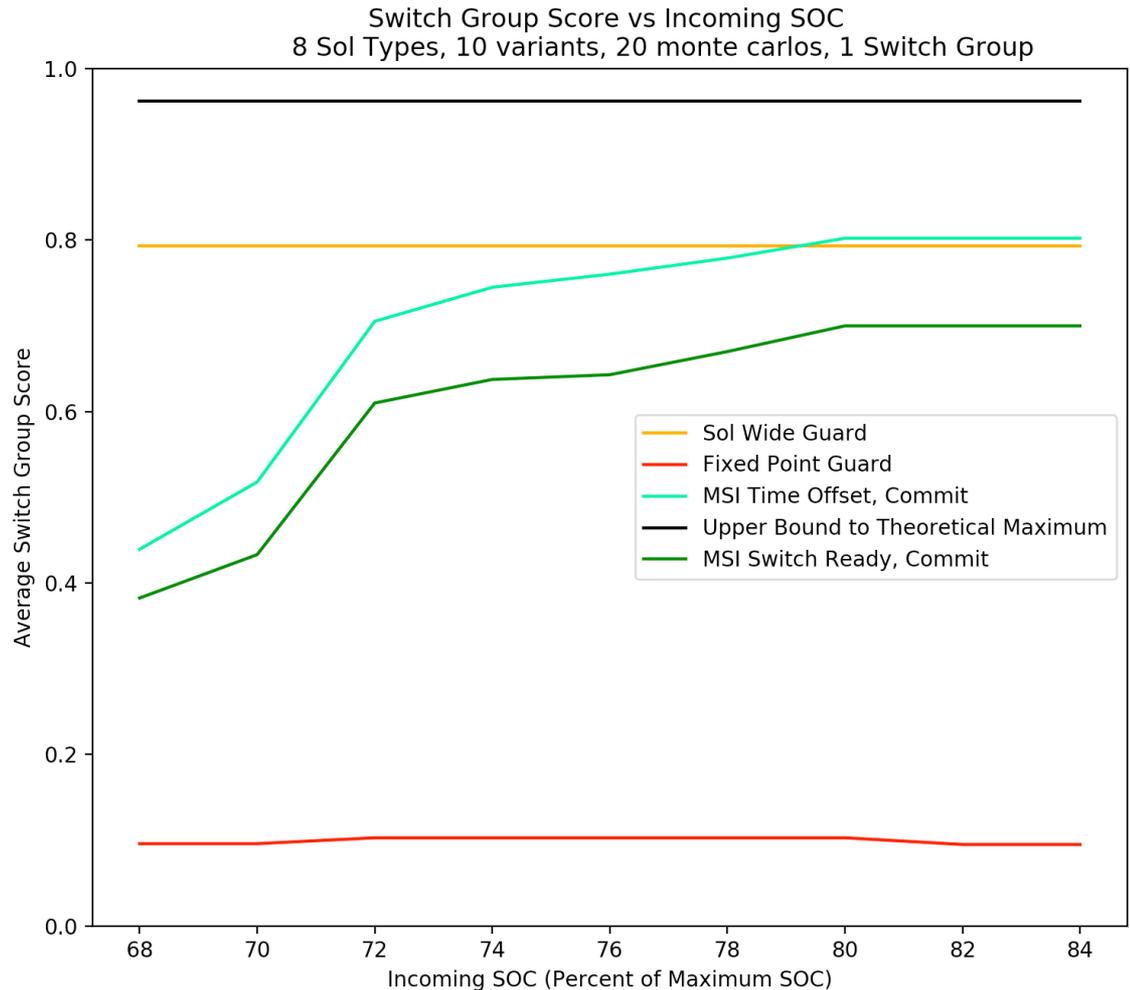
Mandatory Score vs Incoming SOC

- Incoming SOC- energy level at start of schedule
- Also compare to what happens if we always elevate to highest or second highest switch case
 - Elevating to highest performs worst
- **Guard methods and MSI methods that keep nominal switch case committed if it becomes committed during MSI result in all mandatory activities scheduled**
- Vetoing risks activities will not be able to be scheduled in a future invocation



Switch Group Score vs Incoming SOC

- Incoming SOC- energy level at start of schedule
- Upper bound for theoretical maximum switch group score given by *omniscient scheduler*- has prior knowledge of execution durations and is aware of how many resources will be available to schedule higher level switch cases
- **Sol wide guard and MSI with Time Offset and Commit result in highest switch group score**
- Fixed Point results in low switch group score because it checks against a SOC constraint at a specific time regardless of what happens during execution vs when scheduler I actually trying to schedule switch case (as in Sol Wide guard)
- The less time there is to complete MSI invocations, the more likely for switch case to become committed- why Switch Ready performs worse



Future Work

- Analysis of start time windows and dependencies to determine where an activity could be placed without blocking other mandatory activities
- Support for multiple switch groups instead of just one
- Binary search to compute guard for Minimum SSOC Sol Wide Guard
- Further study of MSI
 - Start MSI if activity ends early by at least some amount
 - Evenly space MSI invocations
- Extend approaches to data volume

Conclusions

- Guard methods and MSI methods that keep nominal switch case committed if it becomes committed during MSI result in all activities scheduled
- Sol wide guard and MSI with Time Offset and Commit result in highest switch group score



Jet Propulsion Laboratory
California Institute of Technology

BACKUP SLIDES