Low Pass Filter to Prevent the A330 Incident of Flight CI202 China Airlines A333 at Taipei on June 14th 2020

Carnegie Mellon University
Software Engineering Institute

JUNE 18, 2023

Bjorn Andersson
Dionisio de Niz

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM23-0631

Carnegie Mellon University Software Engineering Institute

Agenda

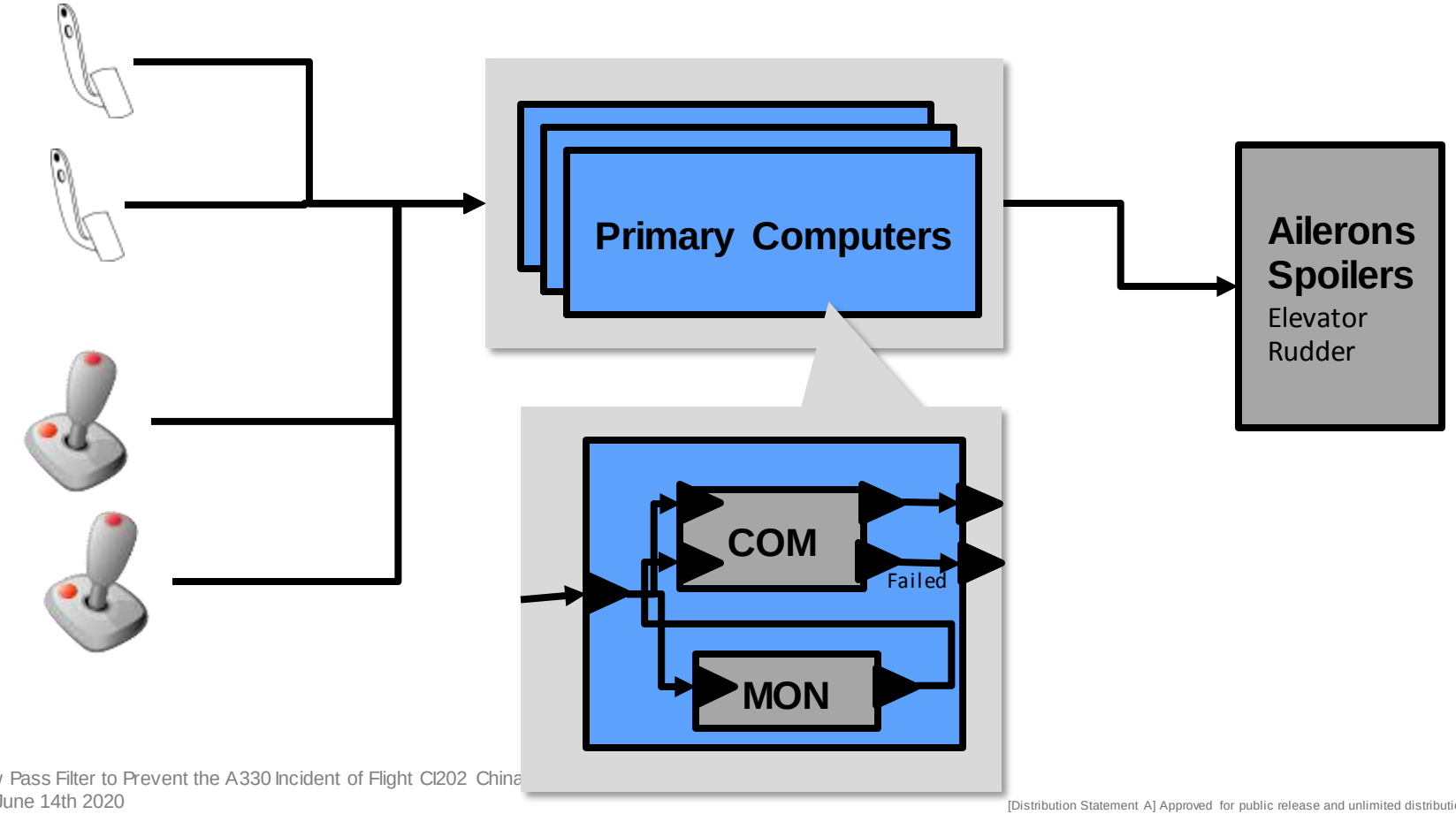
- Recall the incident
- Discuss solution alternatives
- Show our new solution
- Present efficient implementation of new solution

Recall the incident

Carnegie Mellon University Software Engineering Institute

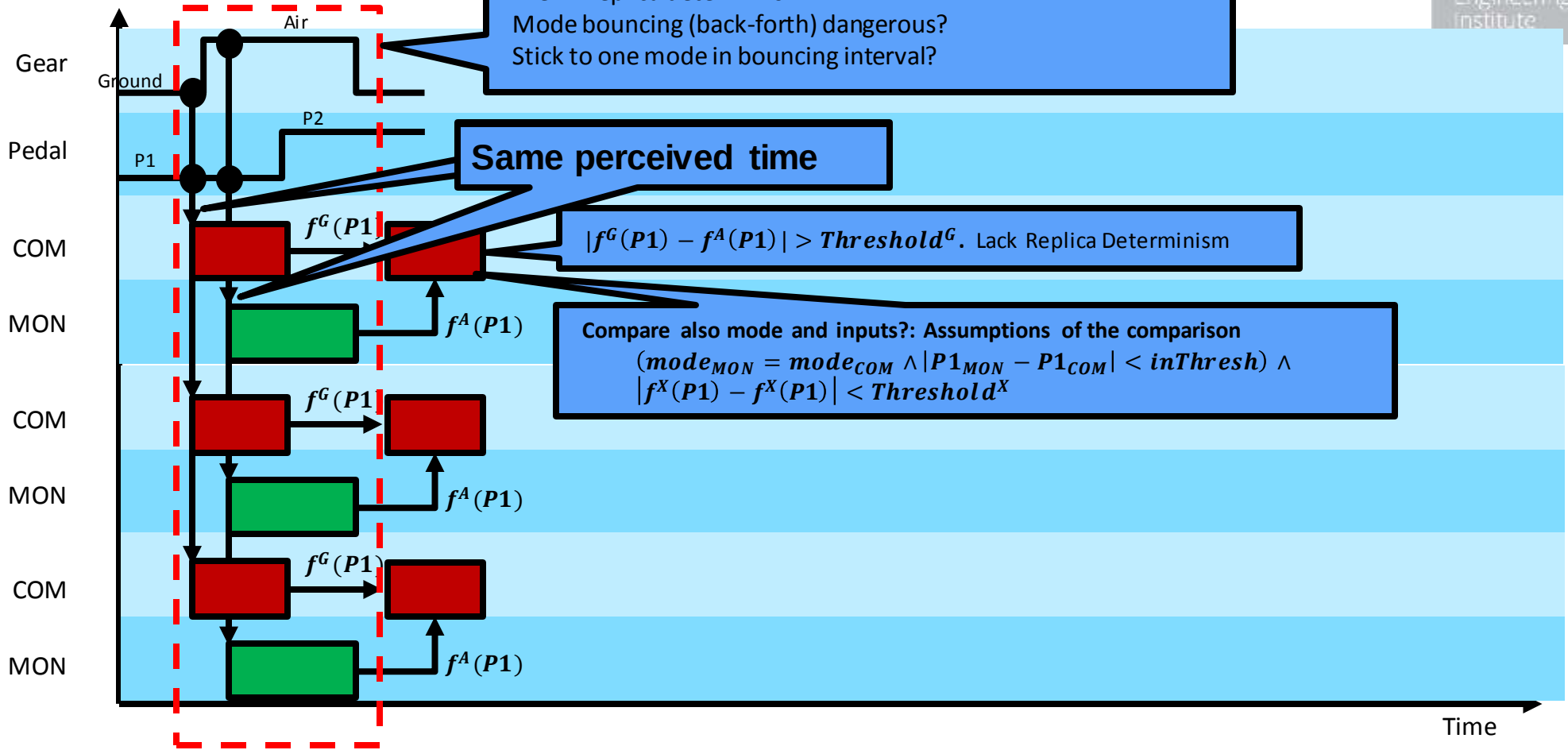
Public information is available at:
https://www.ttsb.gov.tw/media/4936/ci-202-final-report_english.pdf

System Architecture

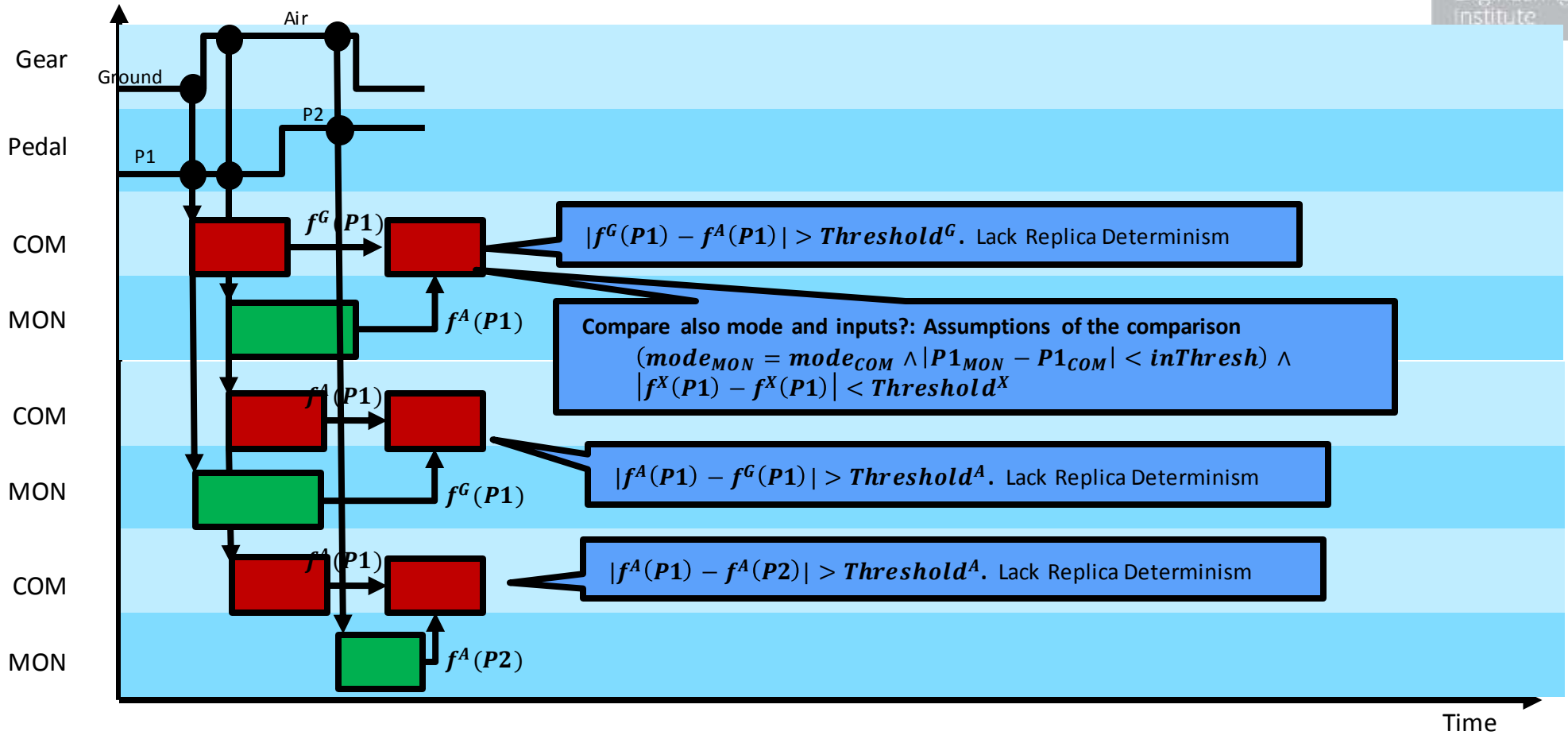


Probable cause

Timelines



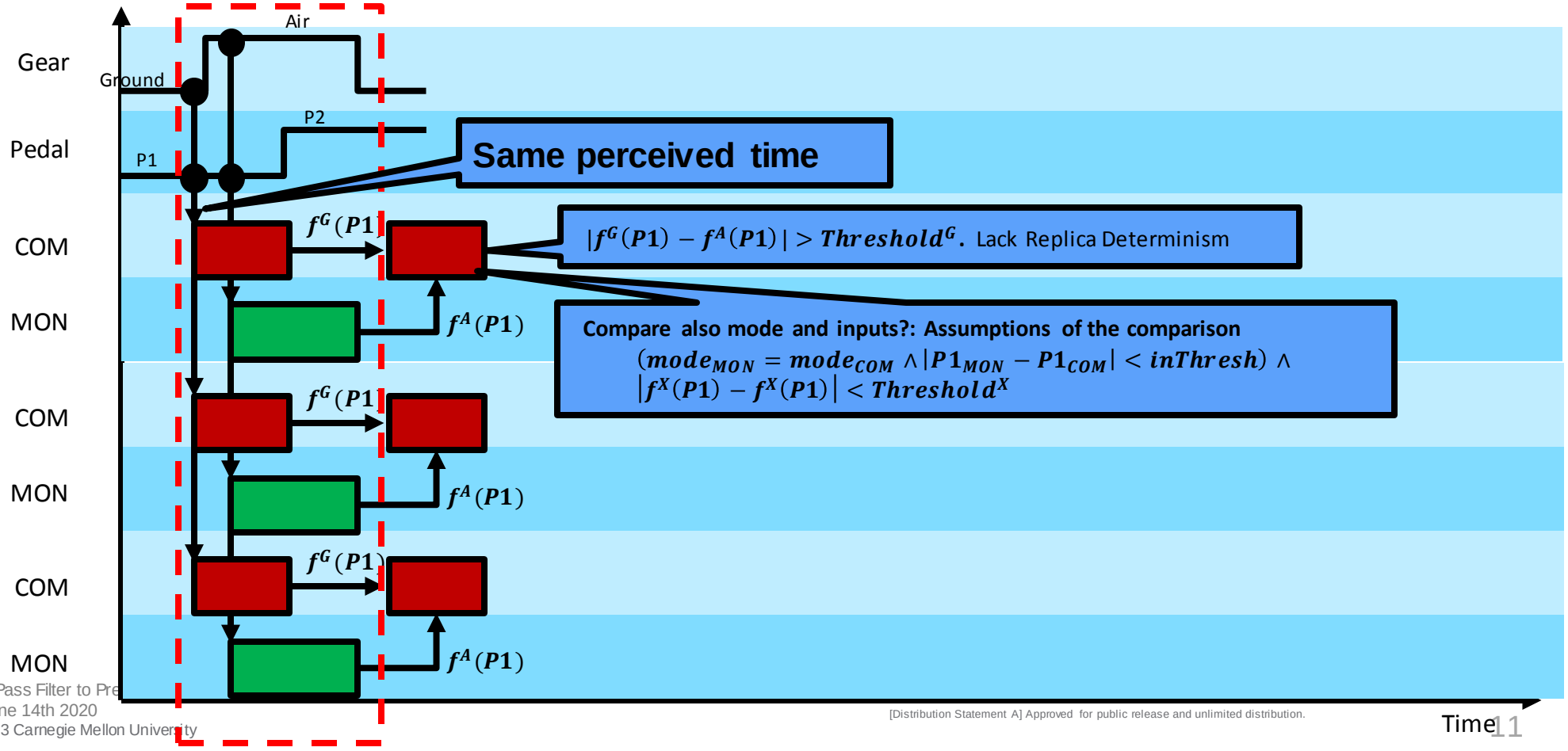
Timelines Alternatives



Solutions that we thought of initially but decided to reject

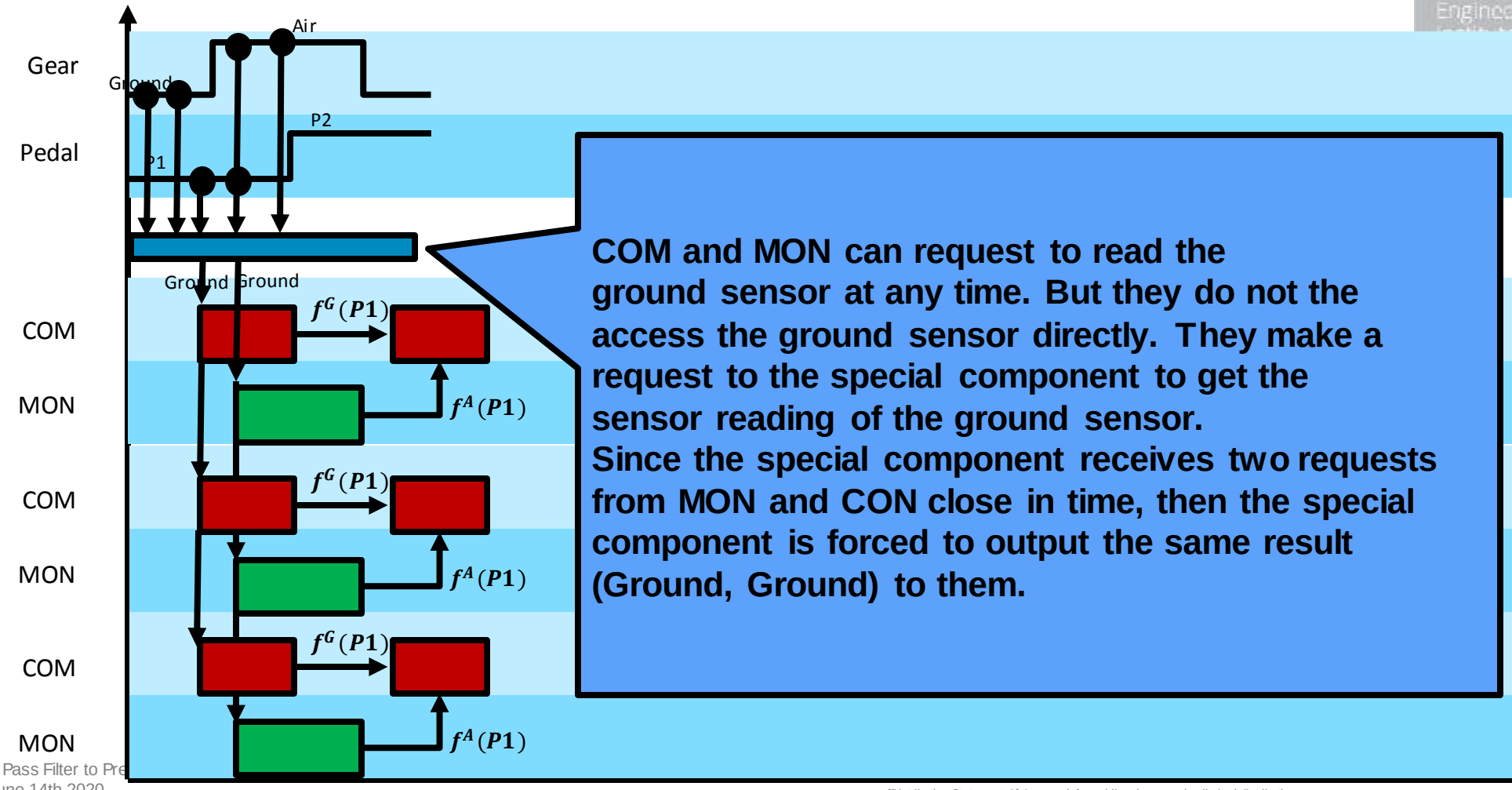
Make sure all computers have synchronized clocks.

- Unfortunately, we can't have perfectly synchronized clocks so it might still happen that we end up with the incident above. Hence, clock synch is not enough.



Introduce a special component that reads sensor that detects ground and delivers this single signal to both CON and MON. If COM and MON requests the single signal sufficiently close in time, then the special component must deliver the same value to those

- This would avoid the issue above. But it comes with the drawback that this special component becomes a single point of failure. This is bad.

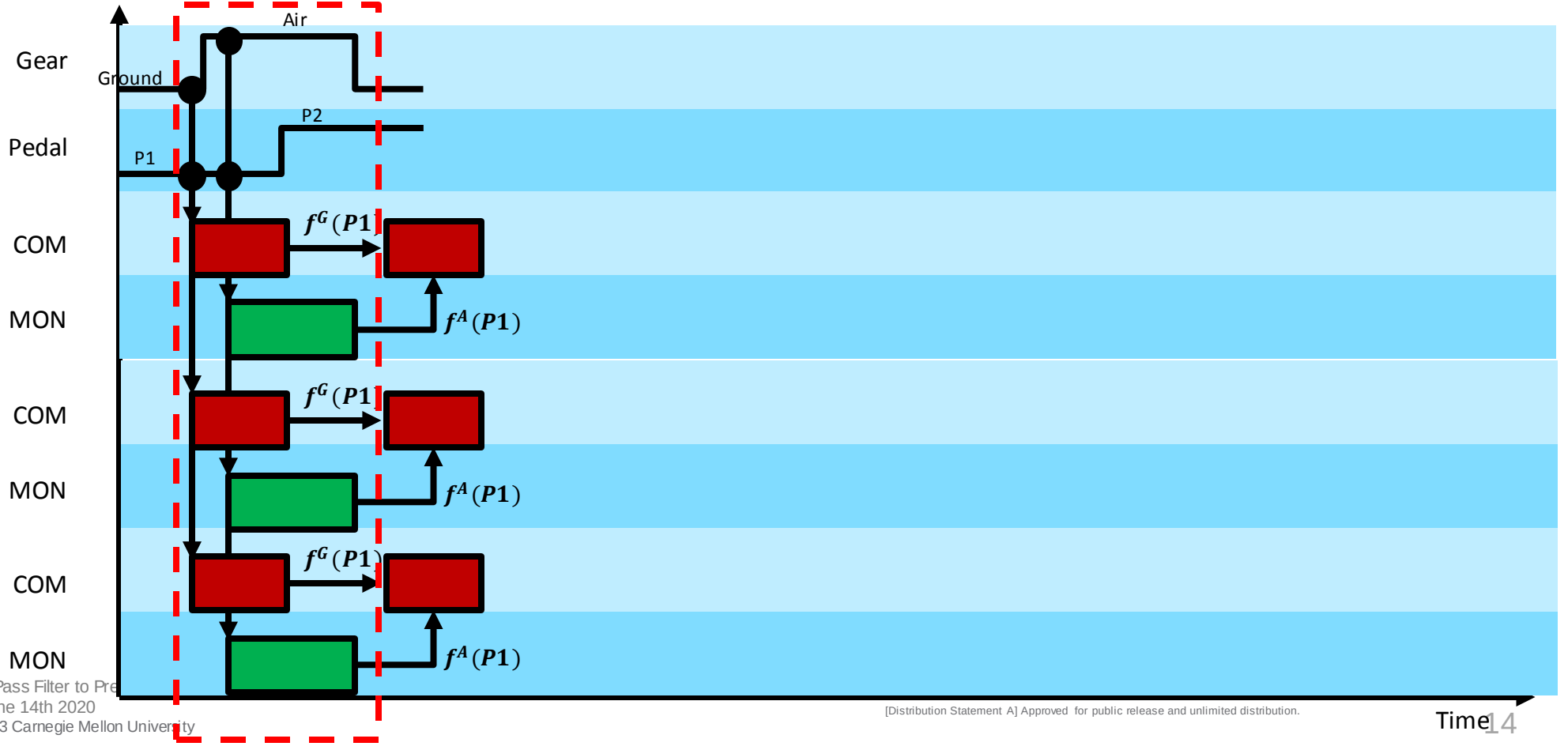


COM and MON can request to read the ground sensor at any time. But they do not the access the ground sensor directly. They make a request to the special component to get the sensor reading of the ground sensor.
Since the special component receives two requests from MON and CON close in time, then the special component is forced to output the same result (Ground, Ground) to them.

Our New Solution

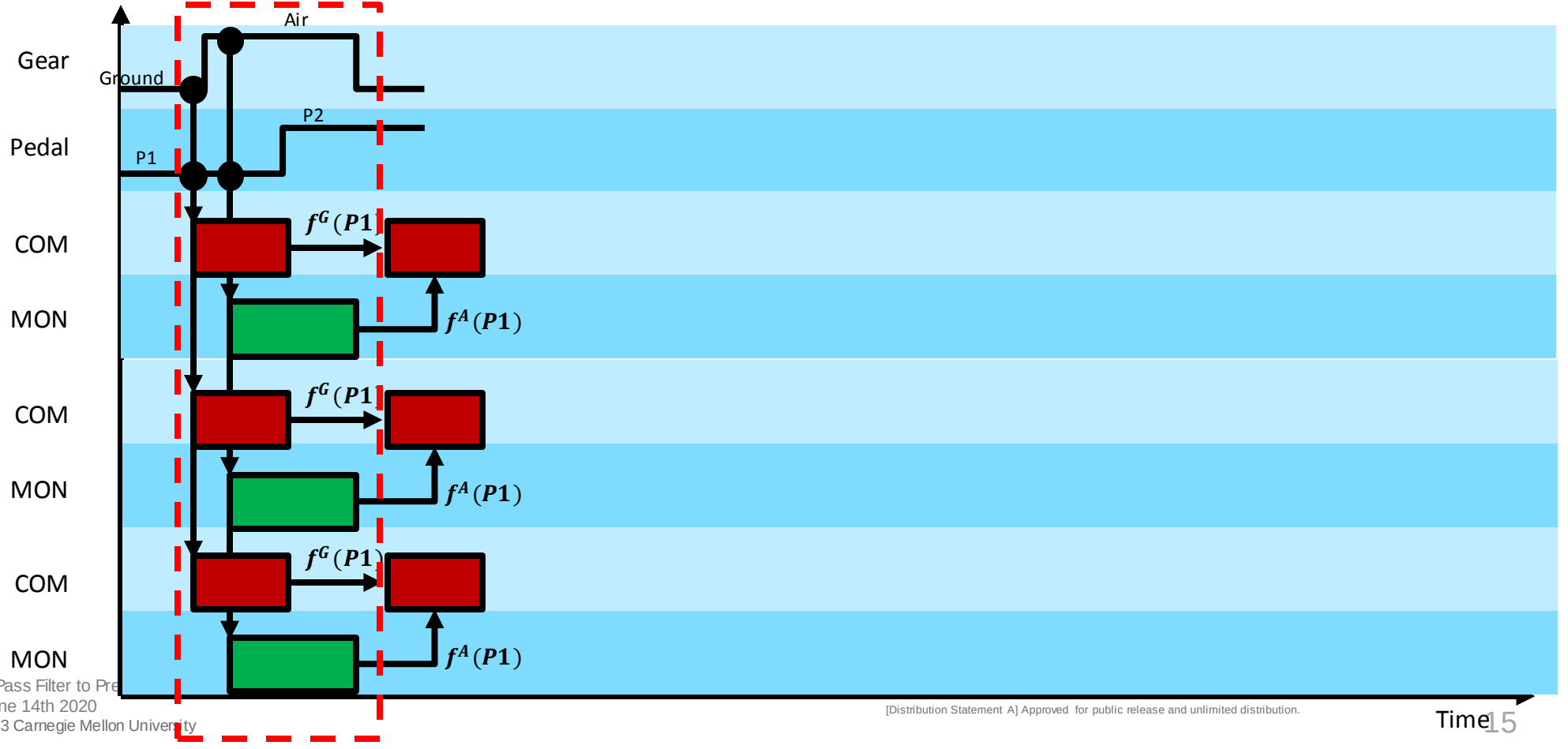
Don't use synchronized clocks. Don't introduce a special component.

- Let COM and MON read ground sensor independently; let them execute independently, let them arrive independently.



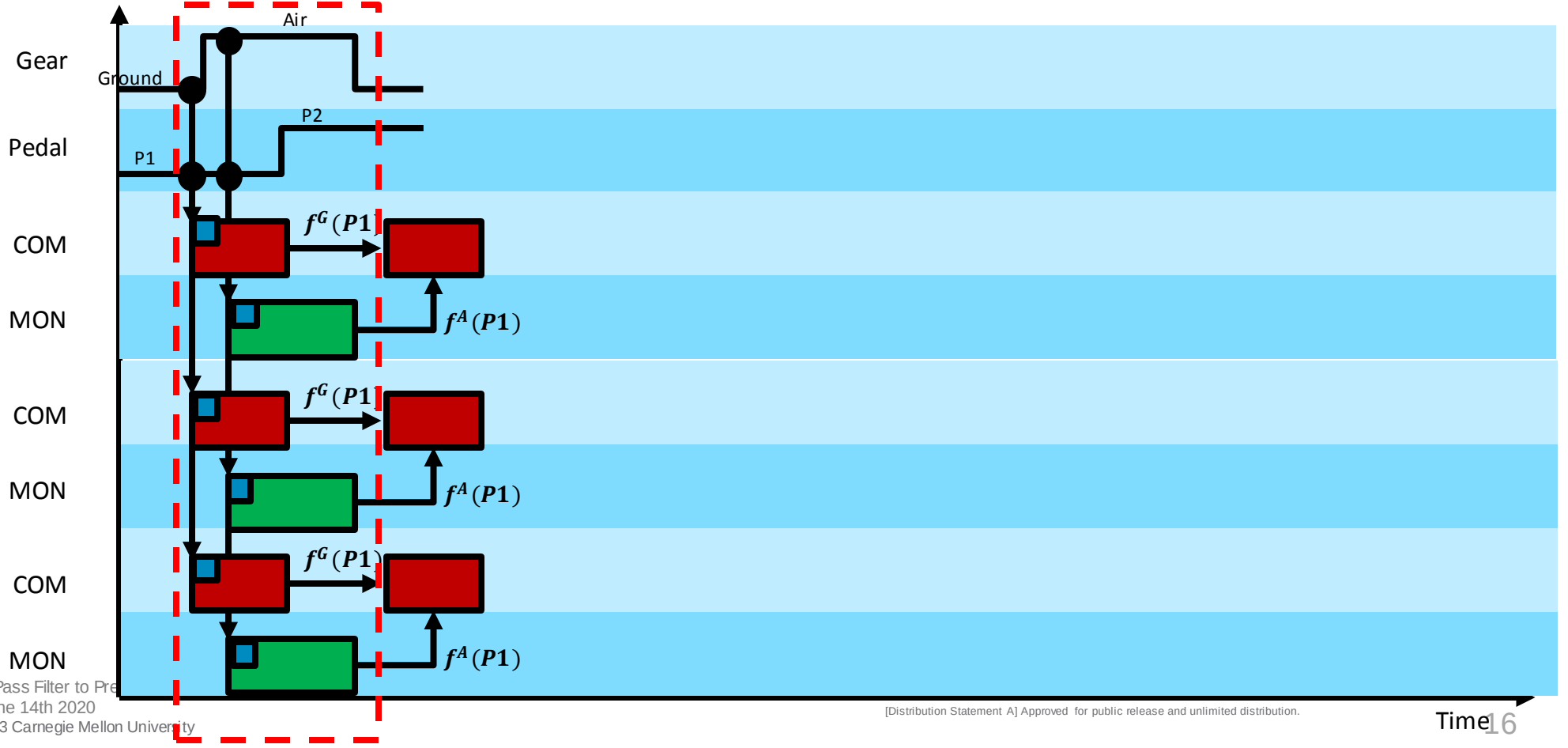
Don't use synchronized clocks. Don't introduce a special component.

- Assume that they execute approximately periodic and that COM and MON have the same period.



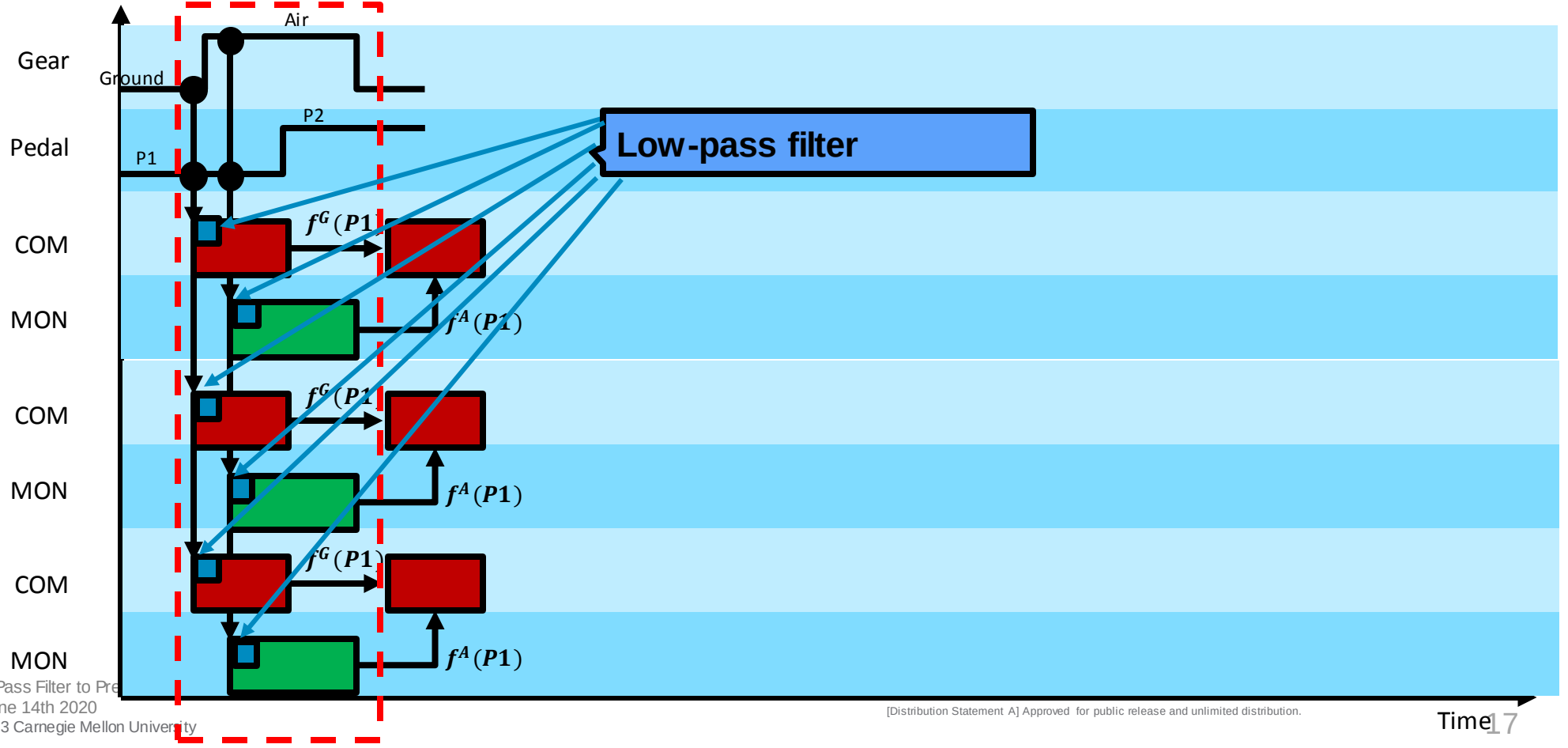
Introduce a low-pass filter in COM and a low-pass filter in MON. These filter are identical.

- Assume that they execute approximately periodic and that COM and MON have the same period.

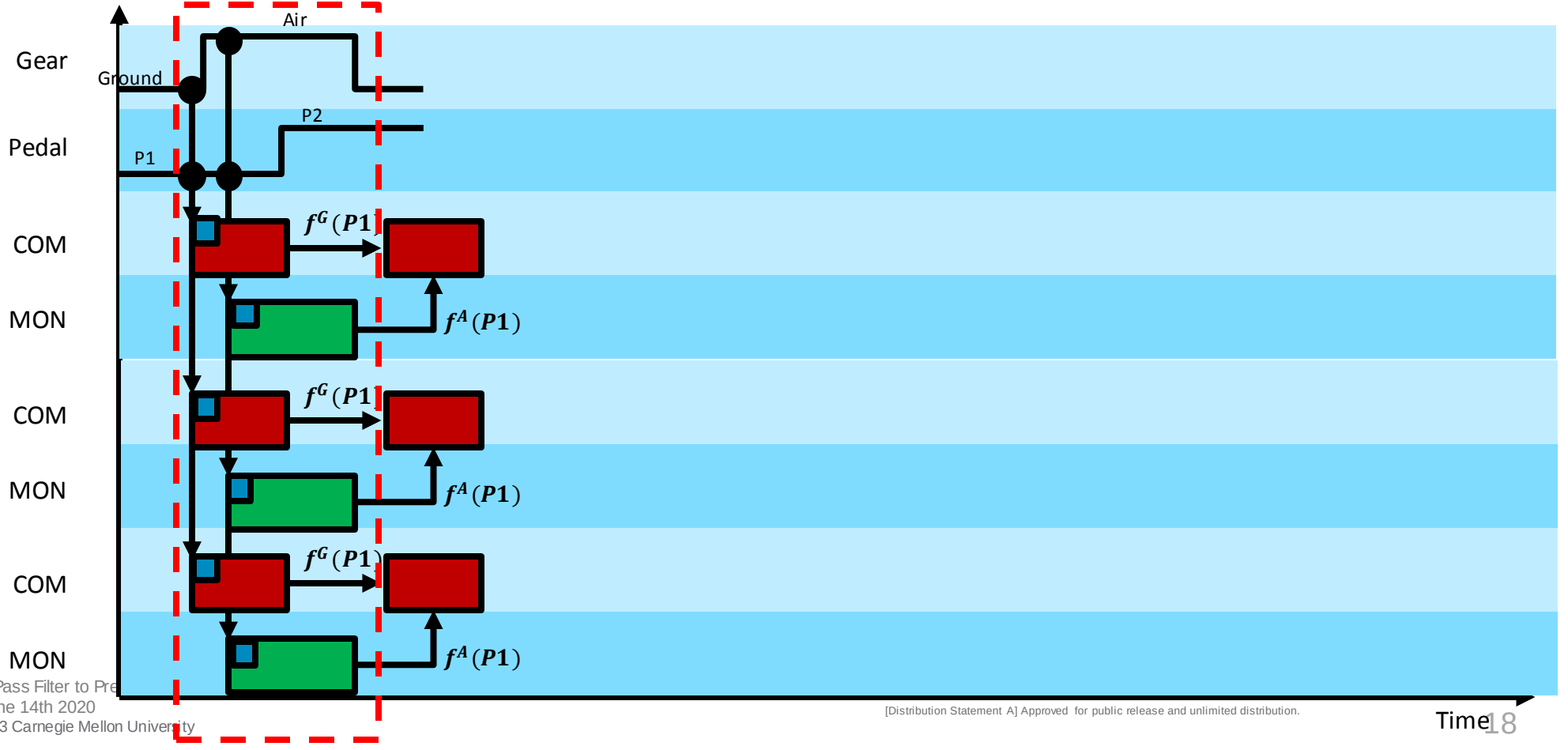


Introduce a low-pass filter in COM and a low-pass filter in MON. These filter are identical.

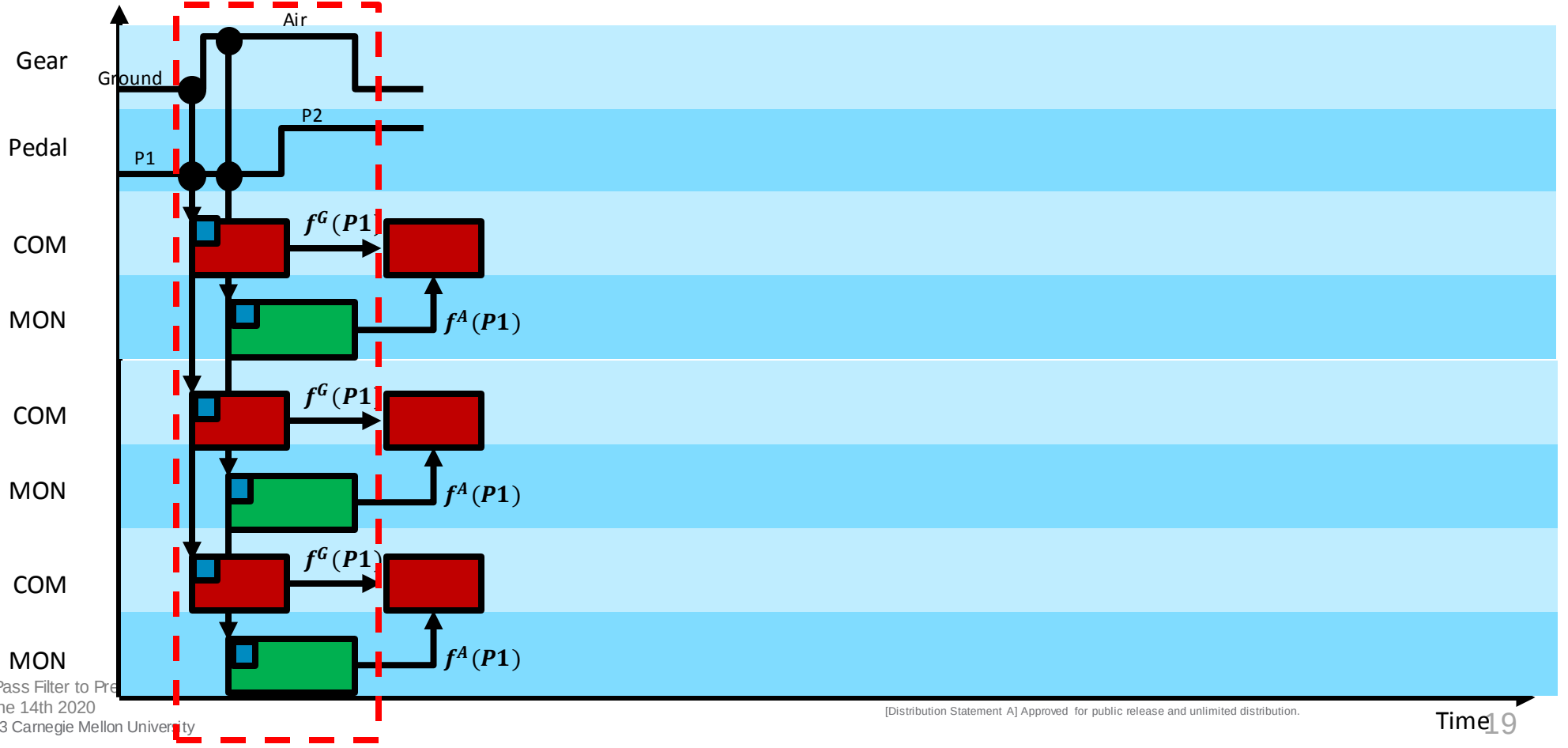
- Assume that they execute approximately periodic and that COM and MON have the same period.



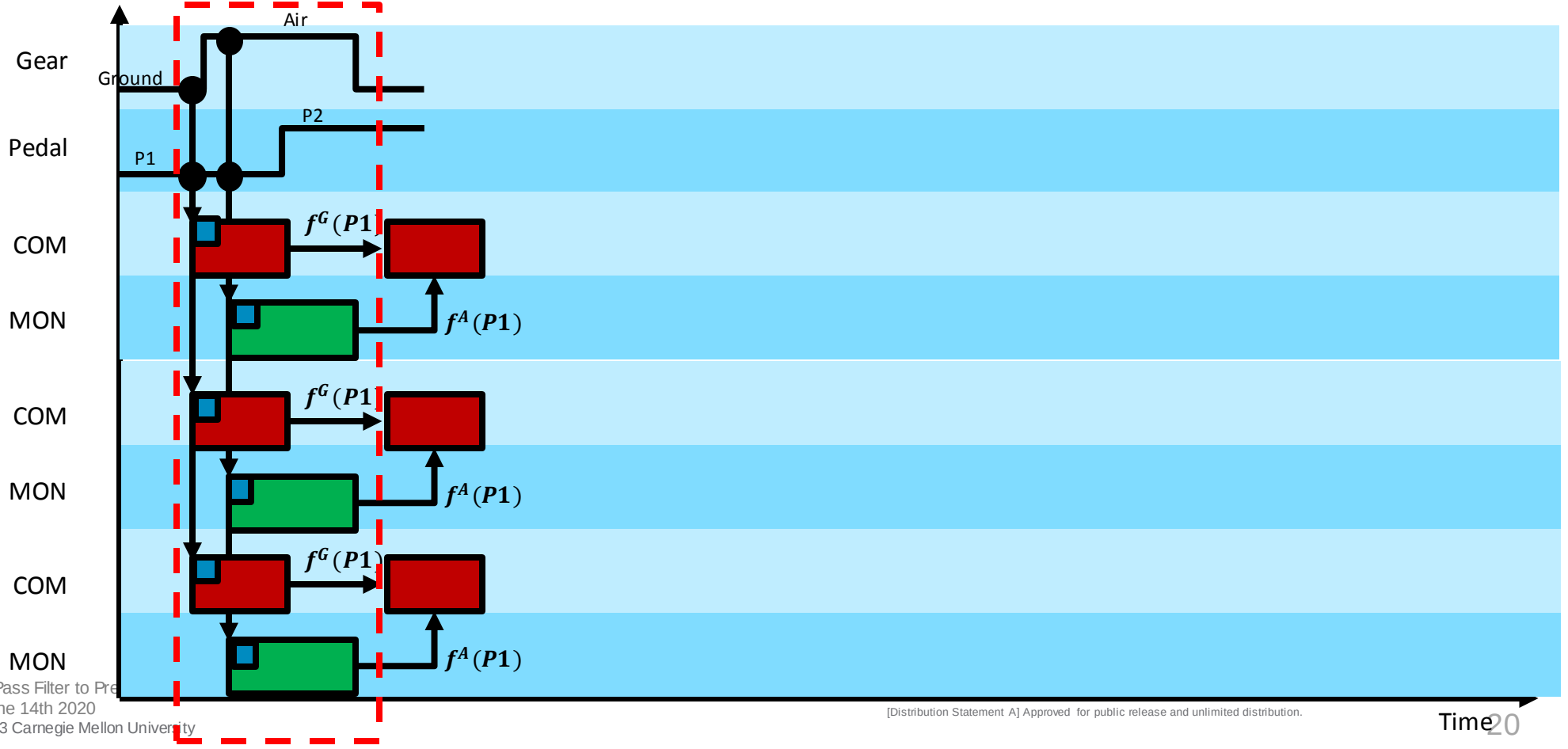
A low-pass filter in COM takes a single Boolean (ground/air) as input and outputs a tuple <m,c> where (i) m is a Boolean, (ii) c is a Boolean, (iii) m indicates mode decision (true means "ground"; false means "air") and (iv) c indicates confidence (True means high-confidence; False means not-high-confidence). Ditto for MON.



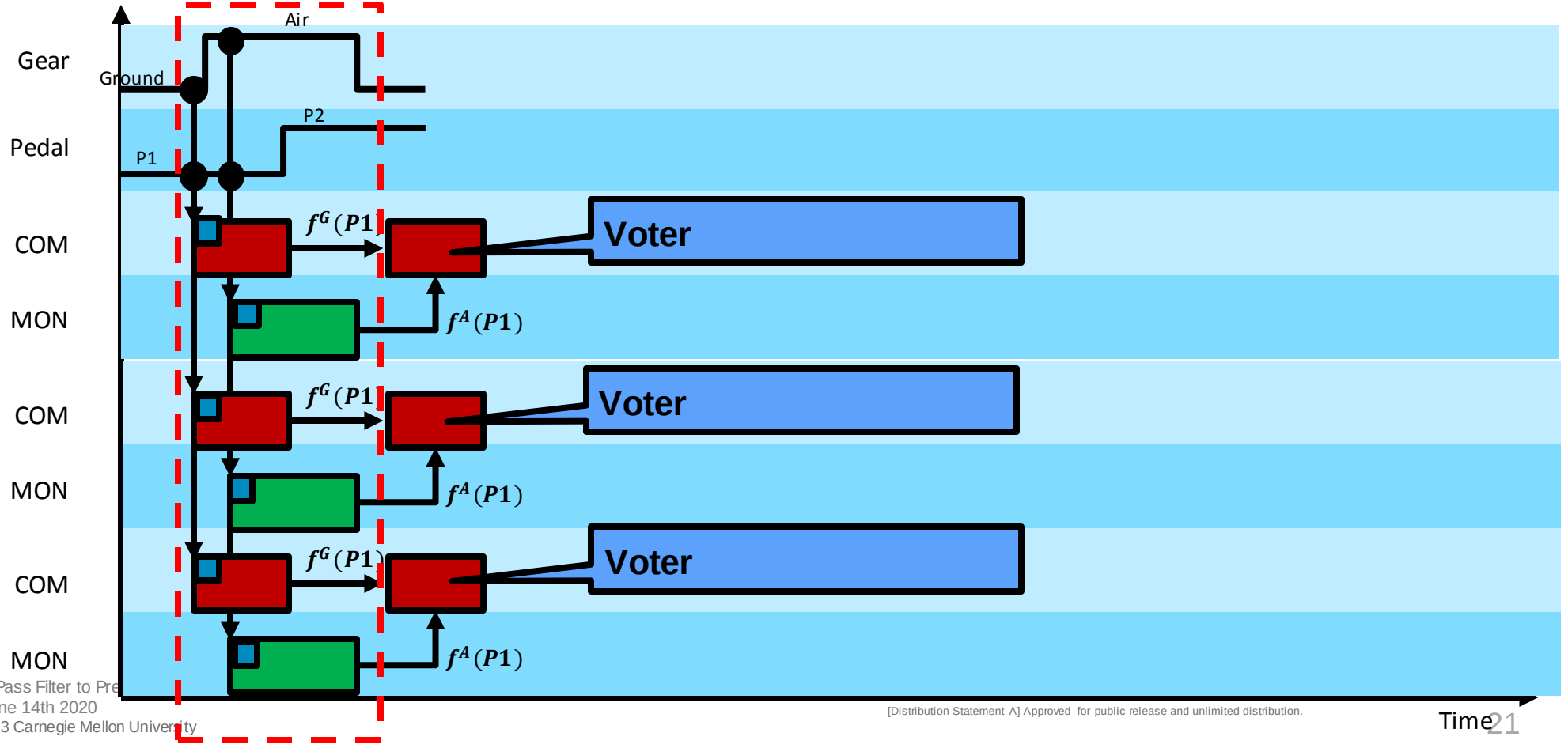
CON outputs f(P1) as shown below and as we have shown before.



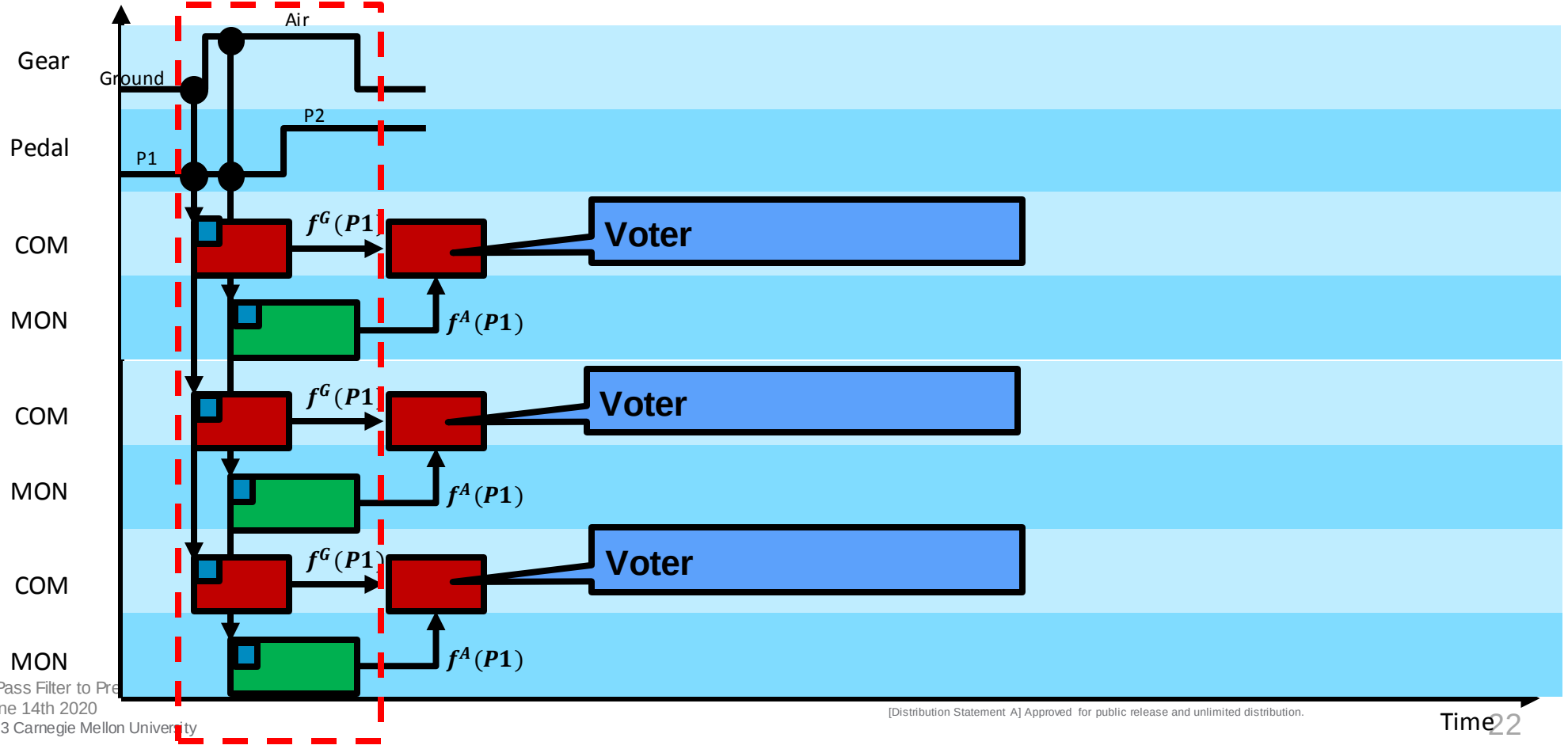
But CON also outputs a confidence signal (Boolean).



CON outputs both f(P1) and confidence signal and they are received by voter.

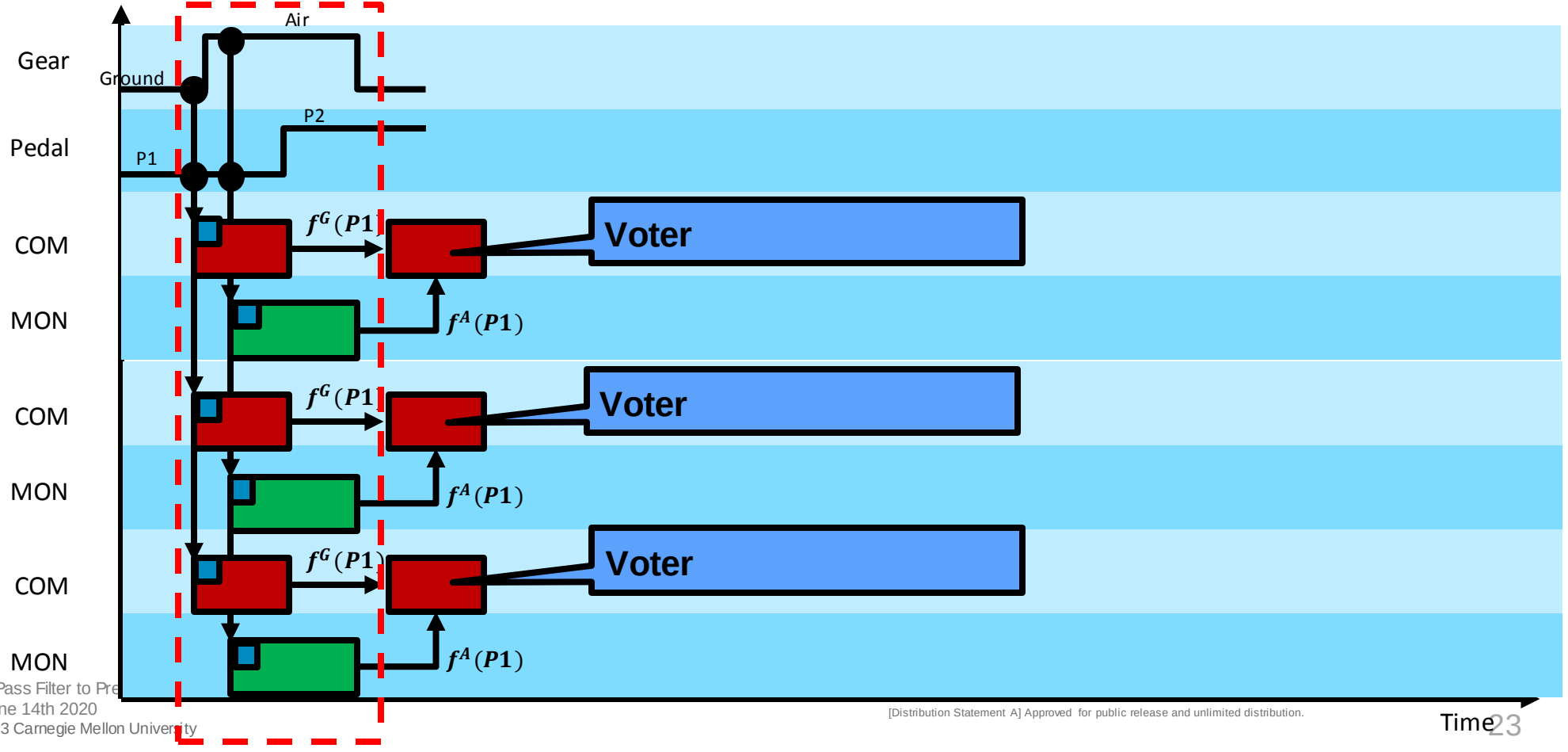


MON outputs both f(P1) and confidence signal and they are received by voter.

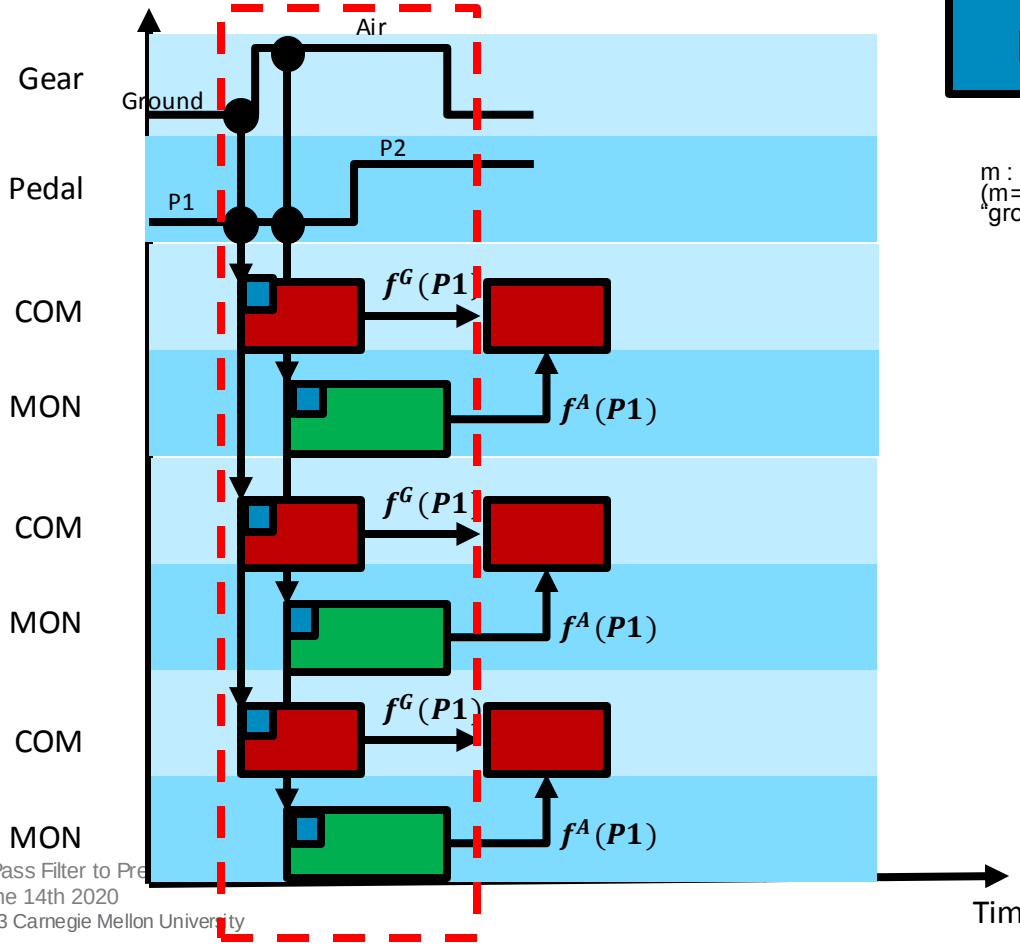


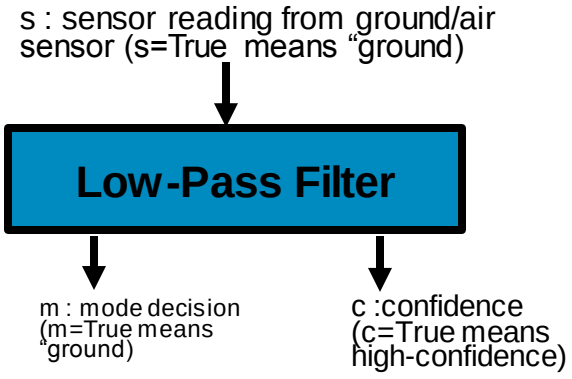
The voter will declare failure if

the f signal from CON and MON are different and the confidence signals from CON and MON are both true.

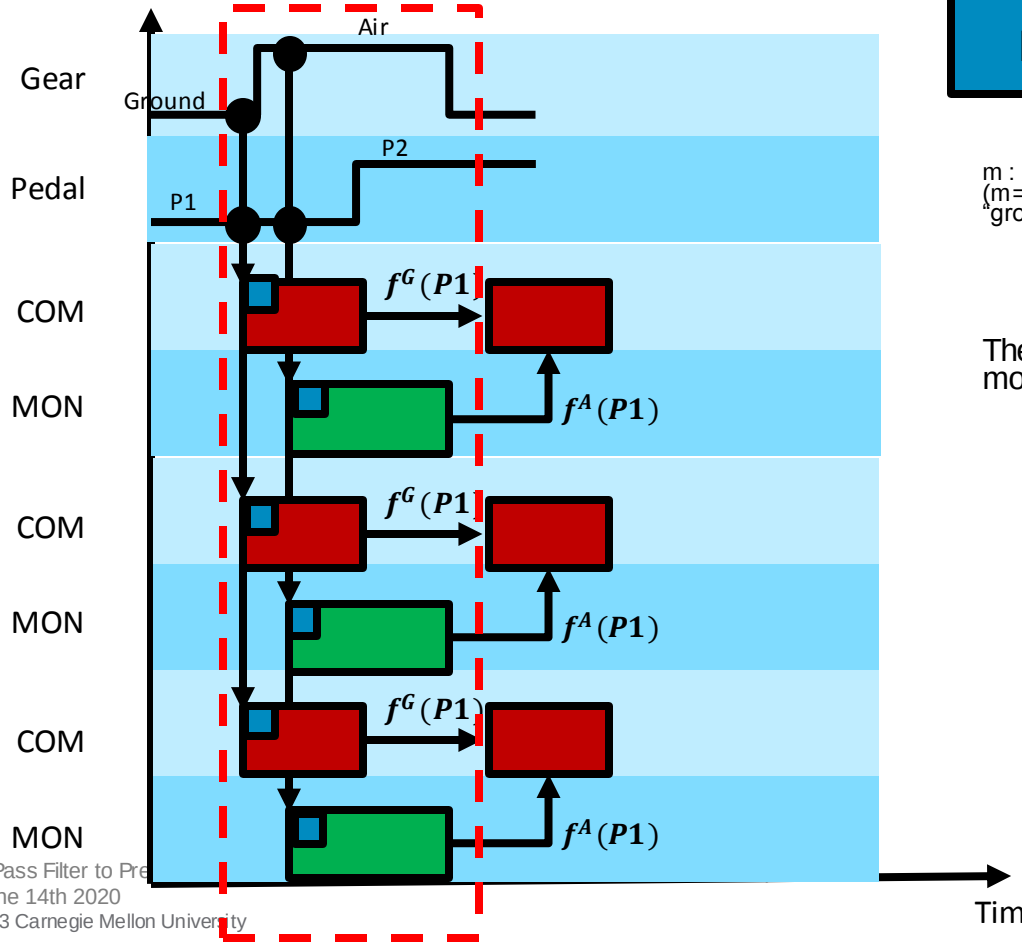


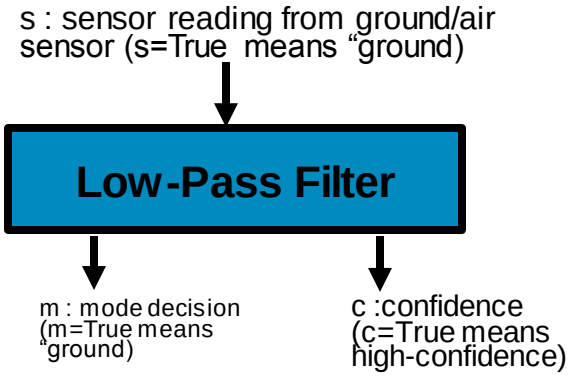
Now, we need to specify the low-pass filter.





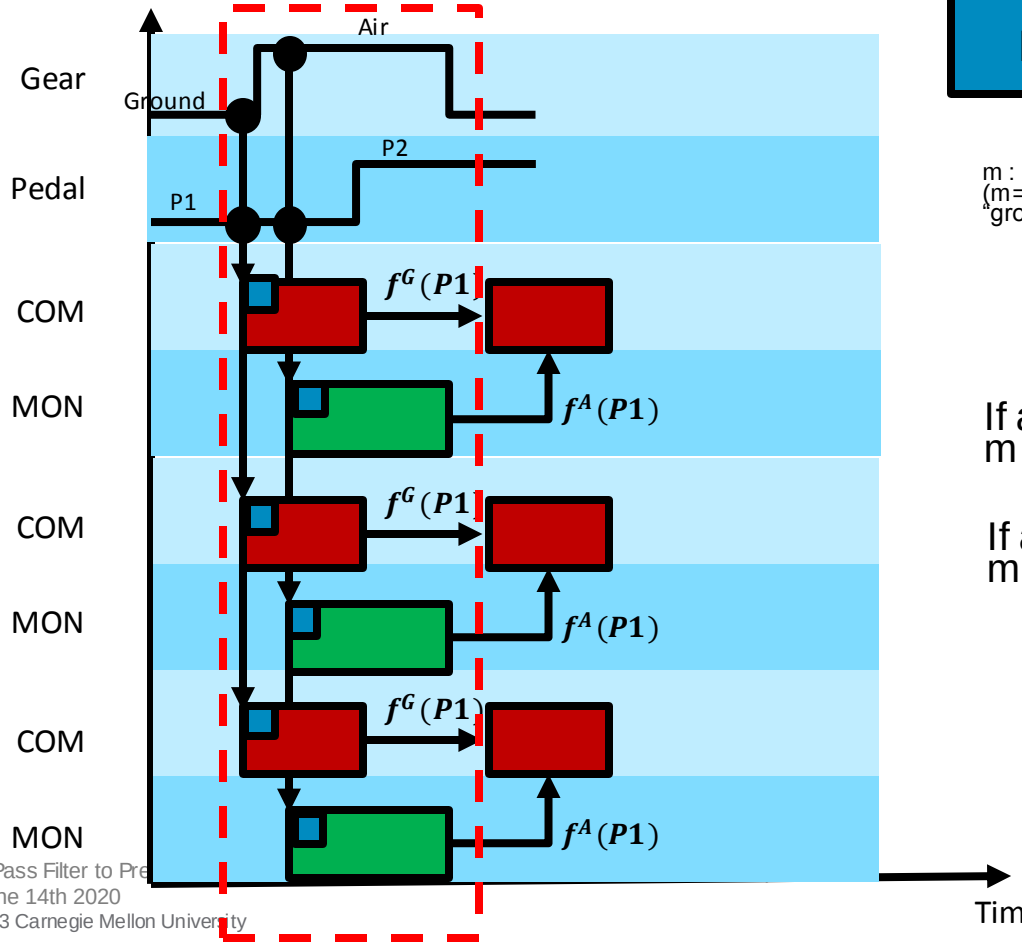
Now, we need to specify the low-pass filter.

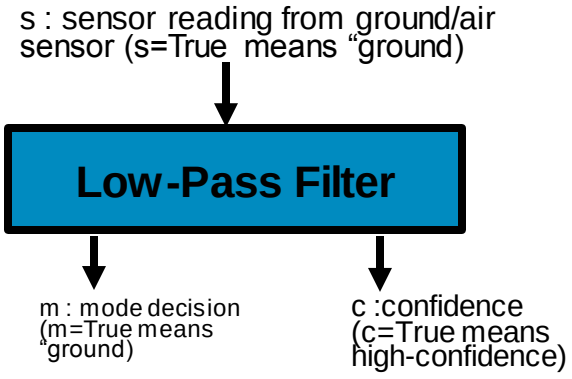




The low-pass filter operates on a sliding window of the most recent WL inputs. For example, WP=5.

Now, we need to specify the low-pass filter.

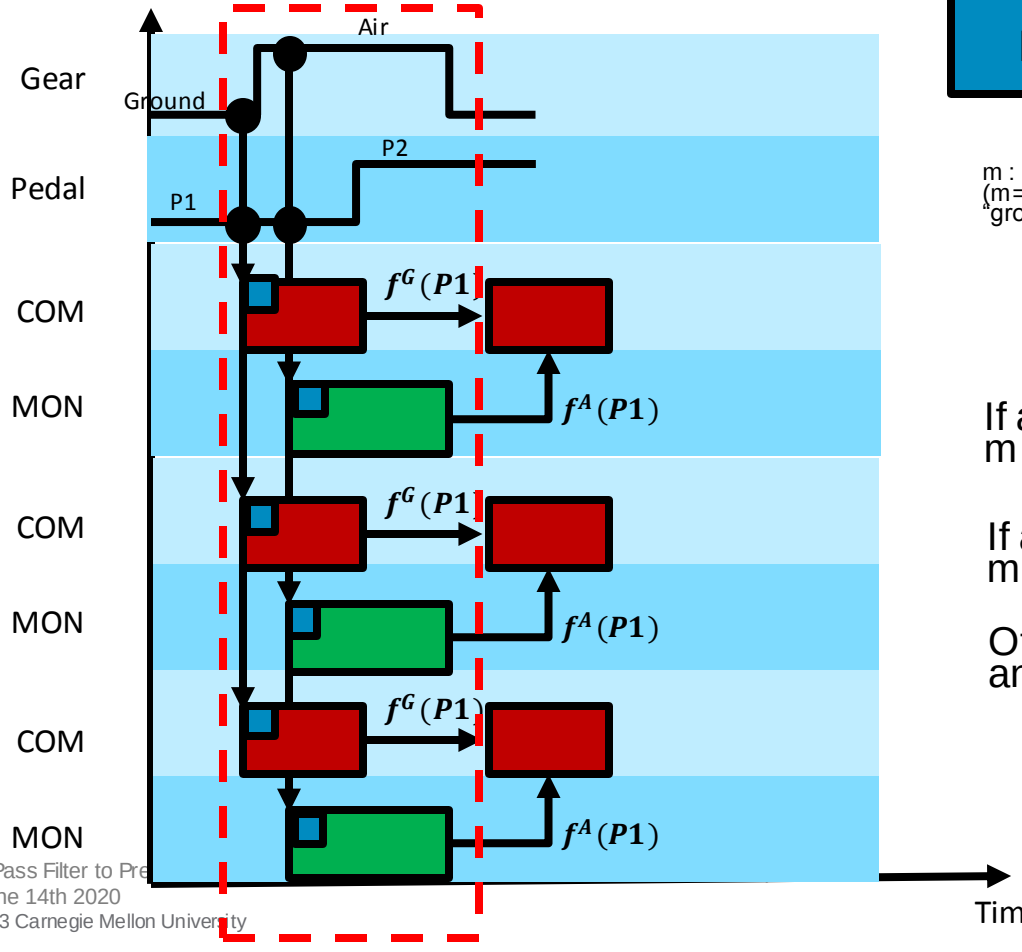


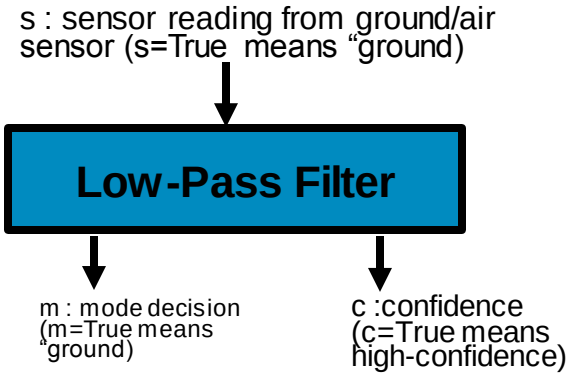


If all WL most recent inputs are True, then m is True and c is True.

If all WL most recent inputs are False, then m is False and c is True.

Now, we need to specify the low-pass filter.



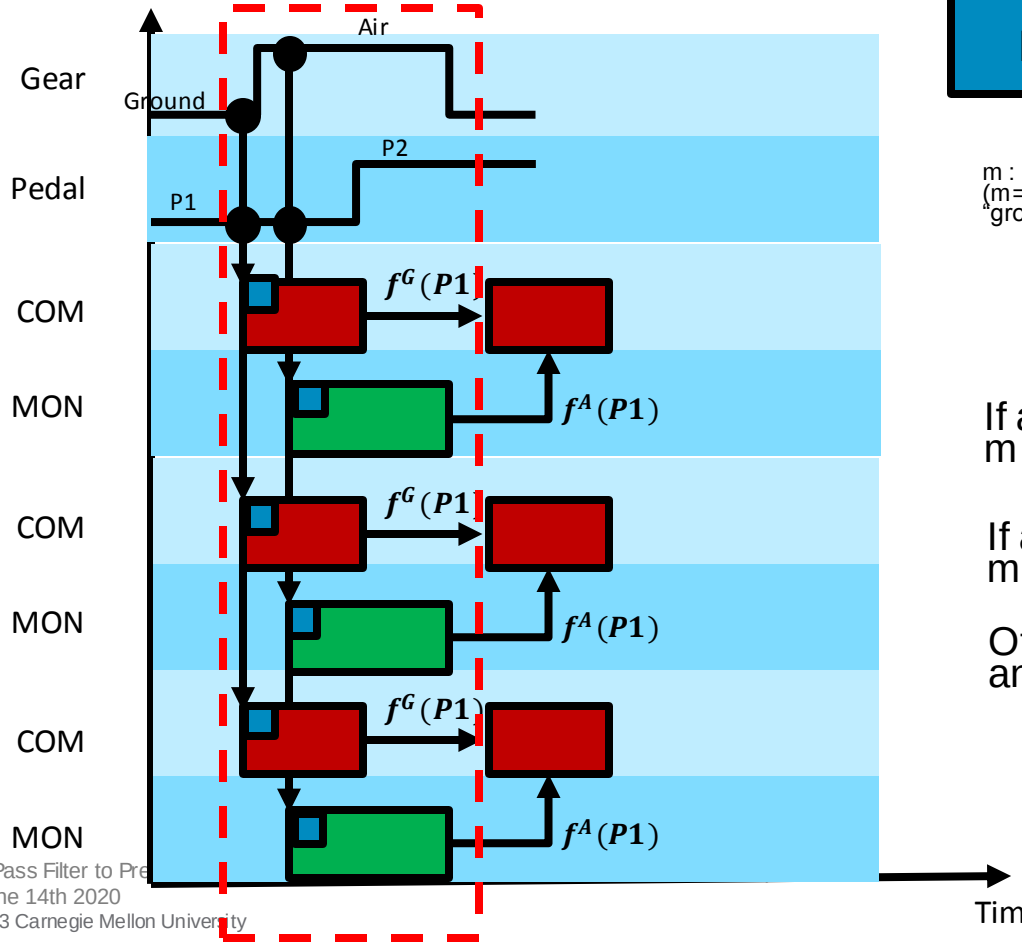


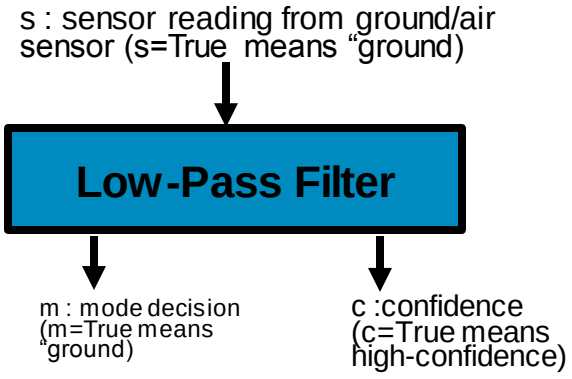
If all WL most recent inputs are True, then m is True and c is True.

If all WL most recent inputs are False, then m is False and c is True.

Otherwise, then m is most-recent-output and c is False.

Now, we need to specify the low-pass filter.





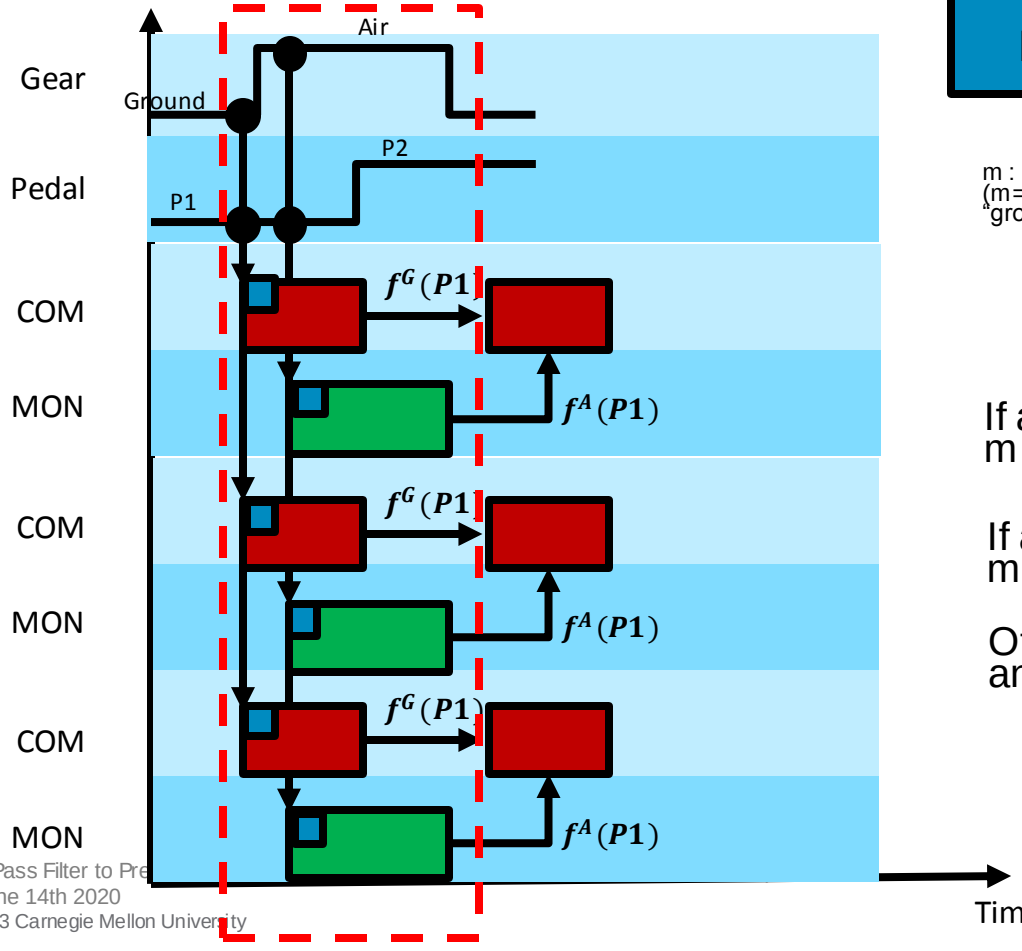
If all WL most recent inputs are True, then m is True and c is True.

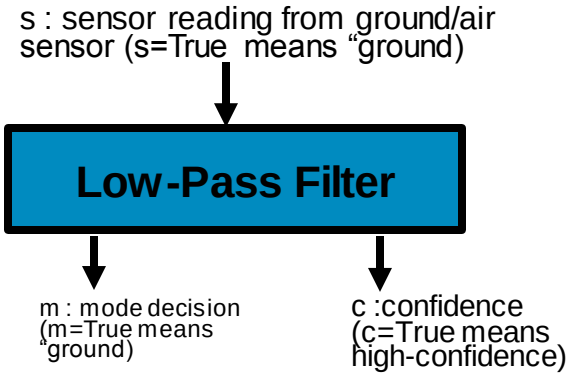
If all WL most recent inputs are False, then m is False and c is True.

Confident during operation

Otherwise, then m is most-recent-output and c is False.

Now, we need to specify the low-pass filter.





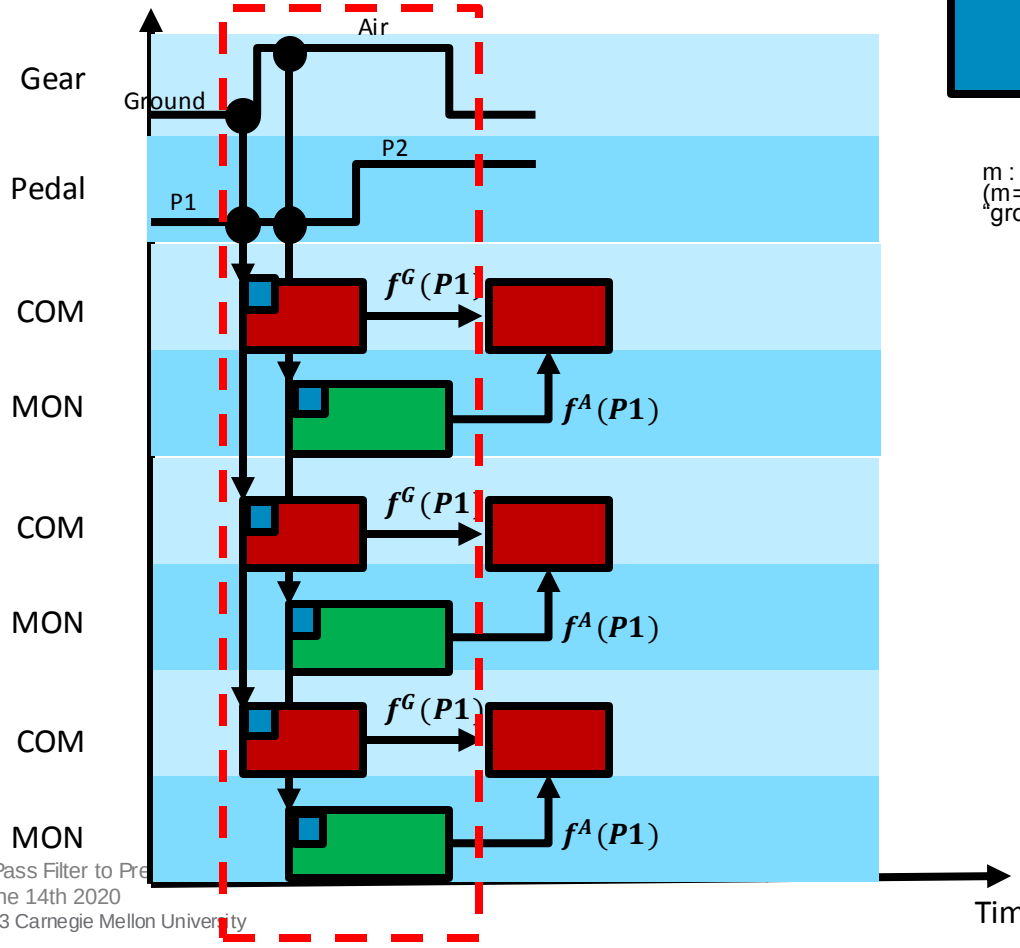
If all WL most recent inputs are True, then m is True and c is True.

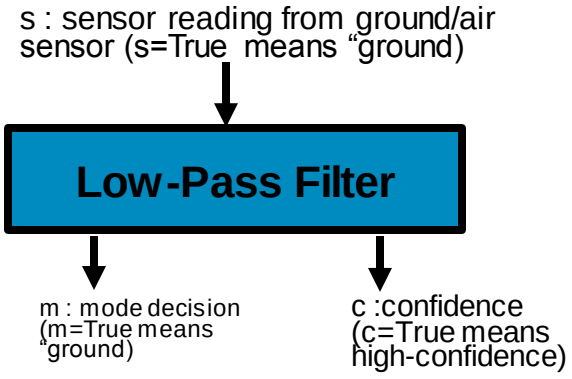
If all WL most recent inputs are False, then m is False and c is True.

Otherwise, then m is most-recent-output and c is False.

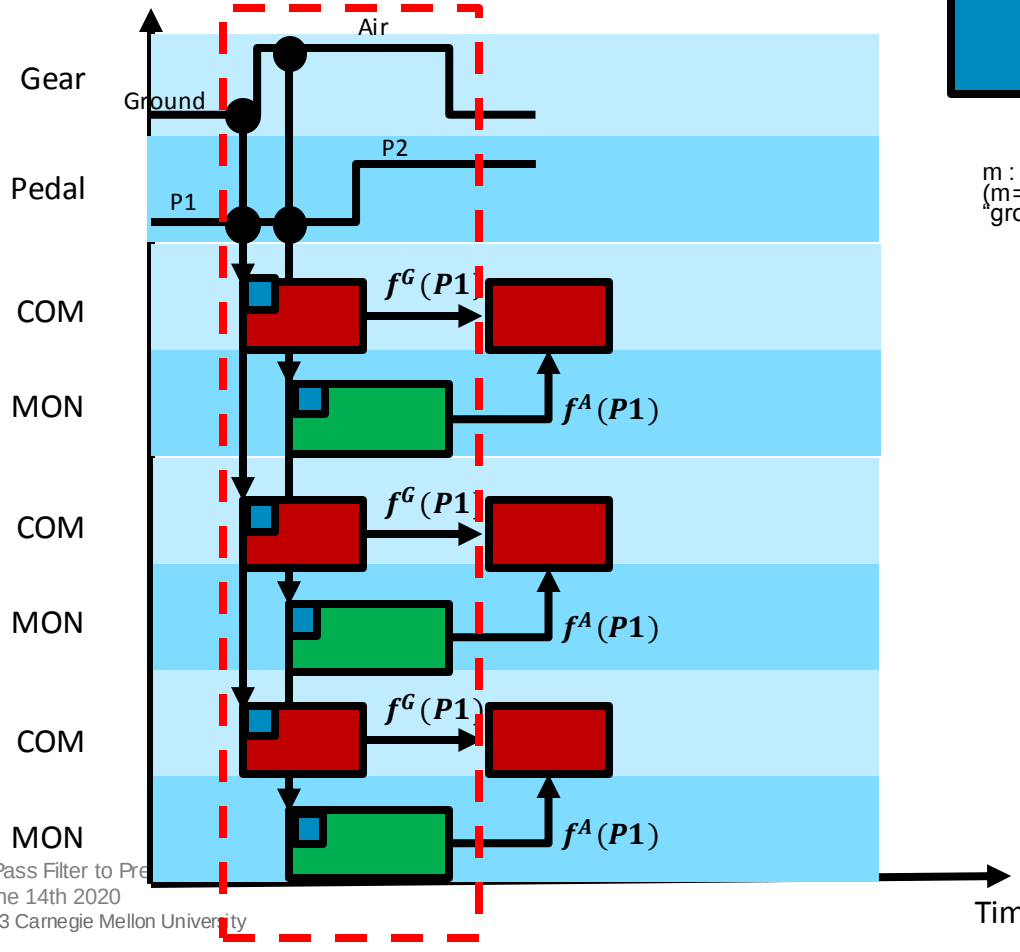
Not confiden during operation

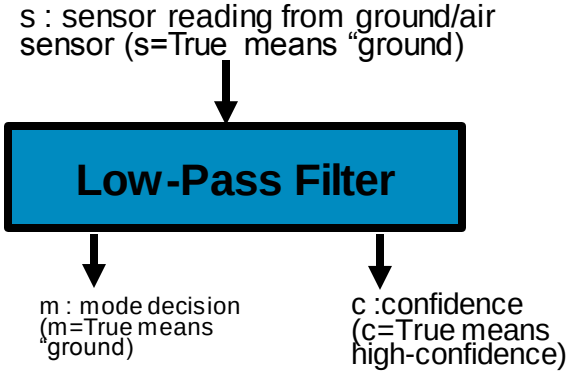
How do we implement this filter efficiently?



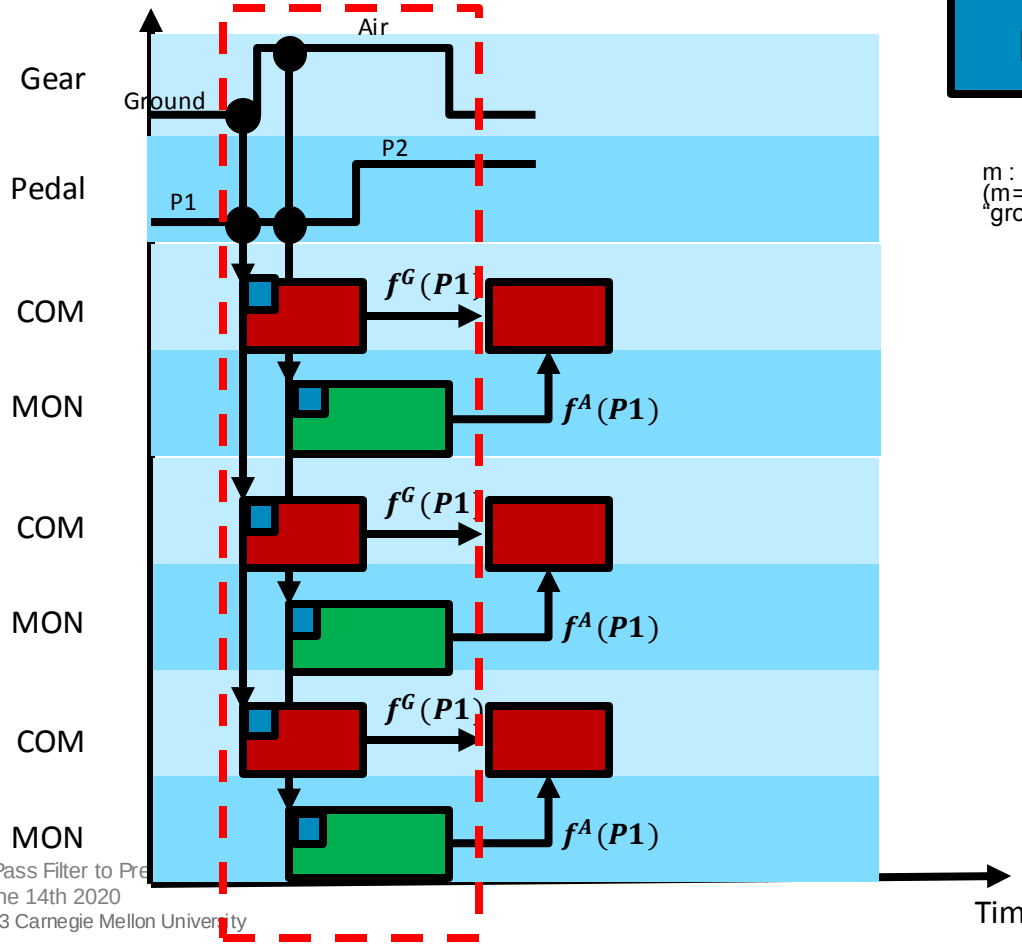


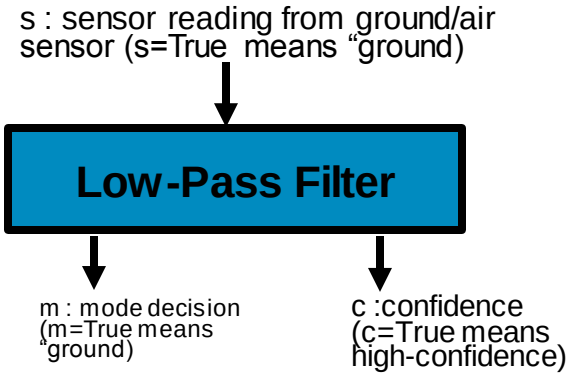
We don't want to have an array with all WL most recent sensor inputs.



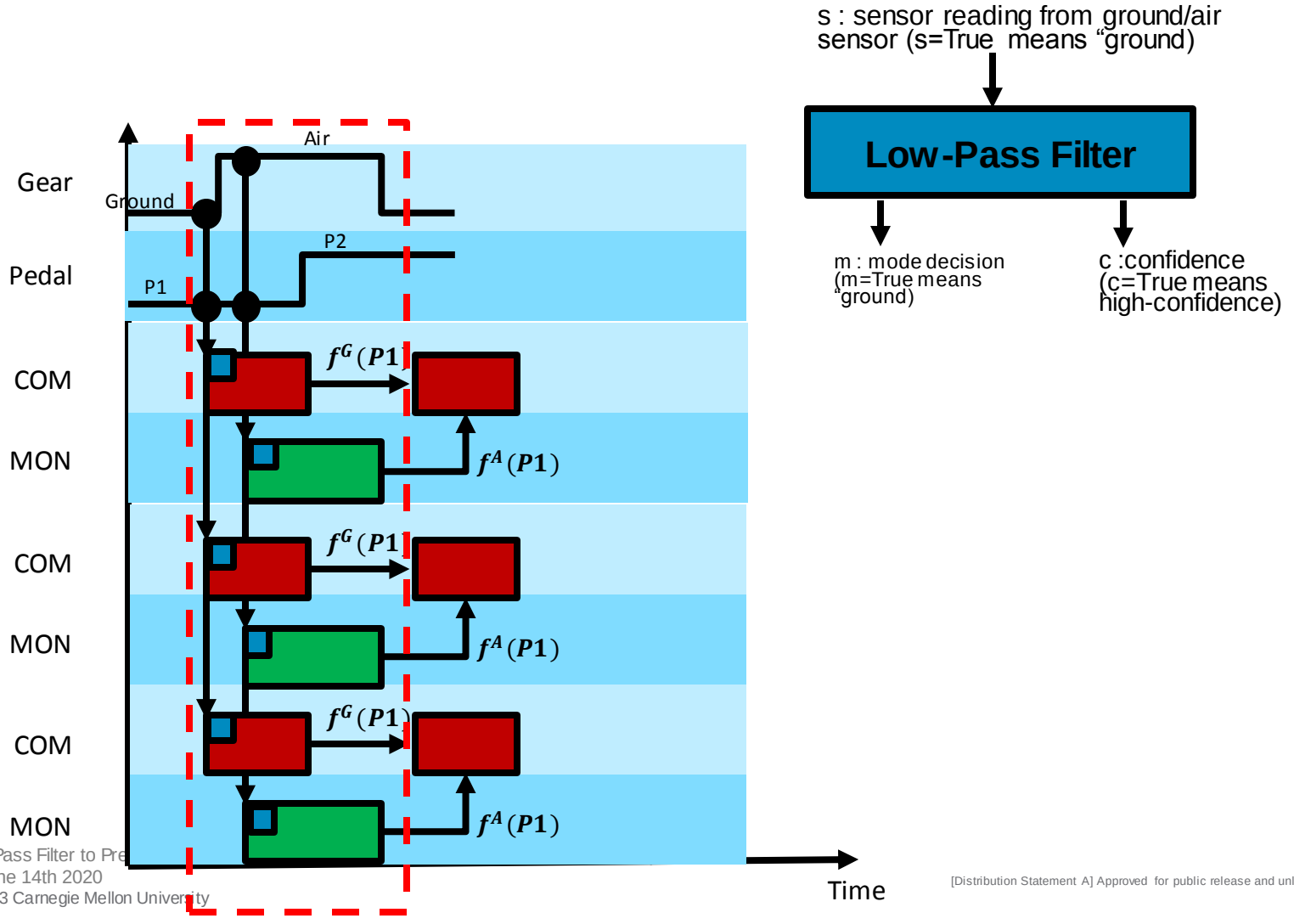


Would consume O(WL) space and would consume O(WL) compute for each input received.



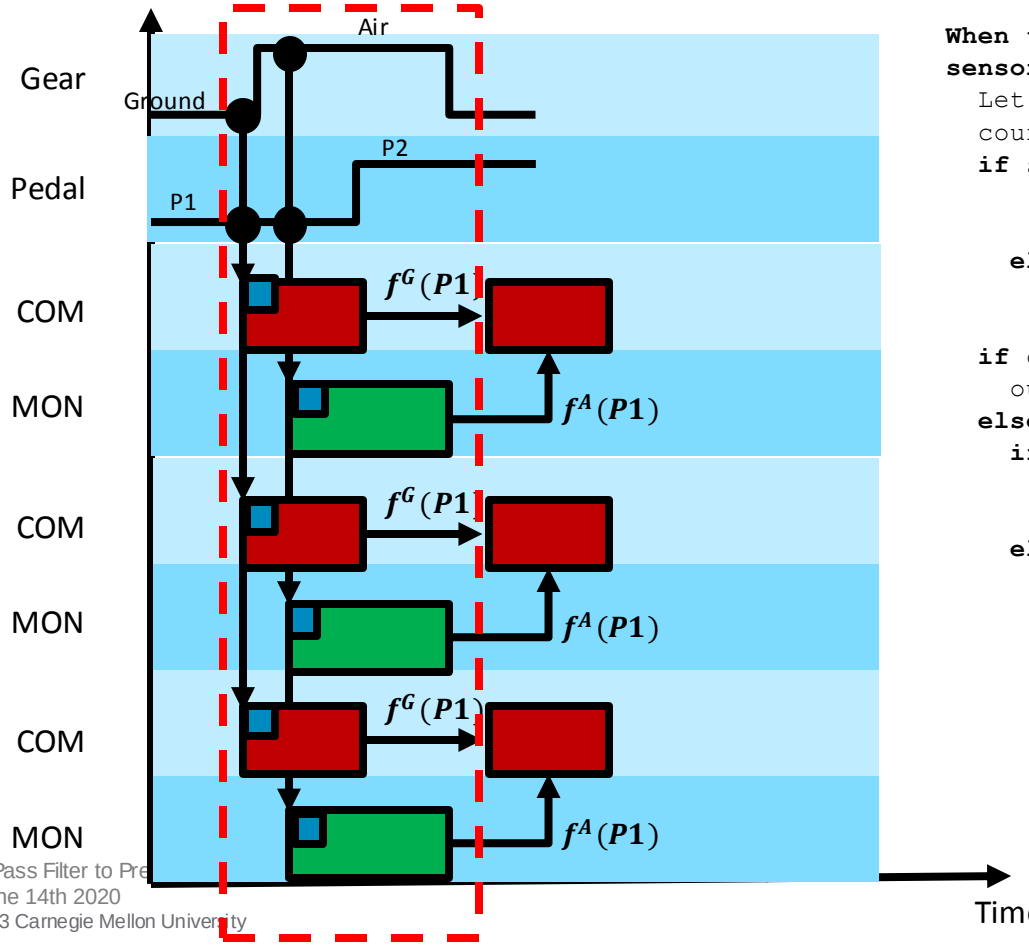


We want to have O(1) space and consume O(1) compute for each input received.



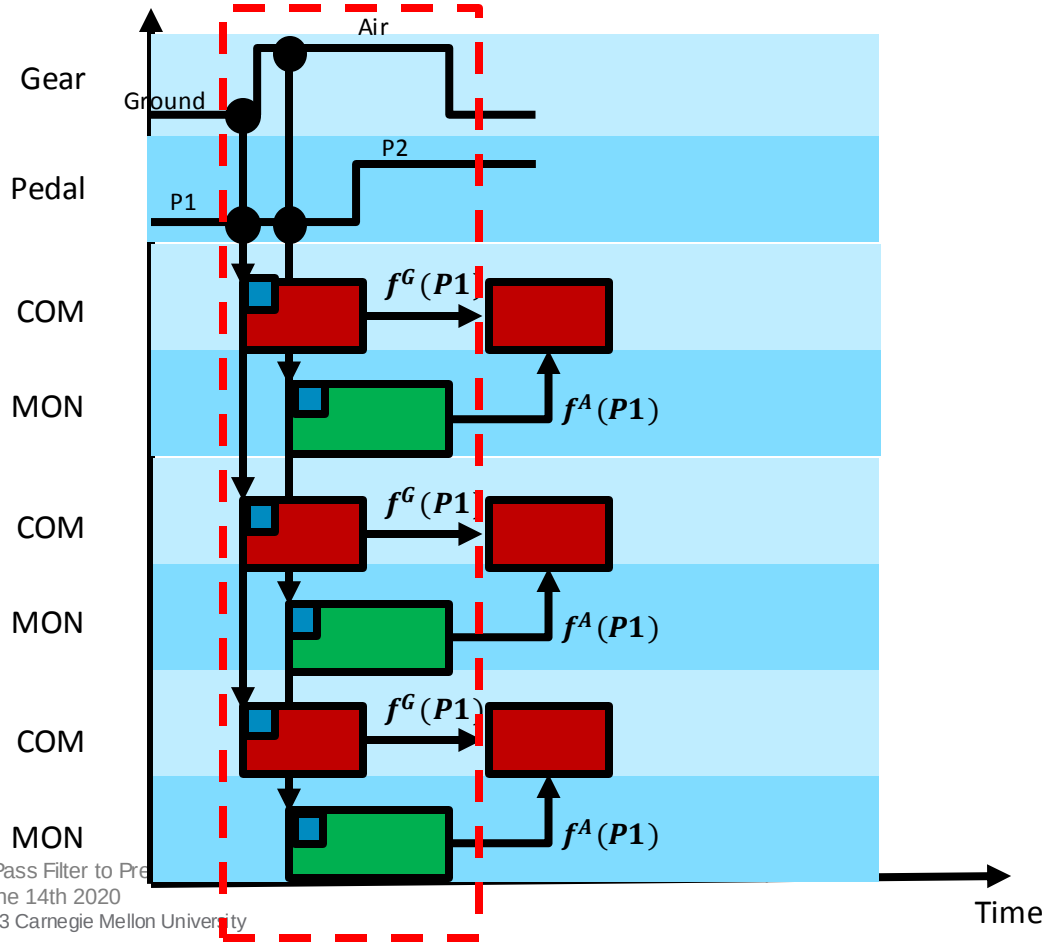
Efficient Implementation

We want to have O(1) space and consume O(1) compute for each input received.

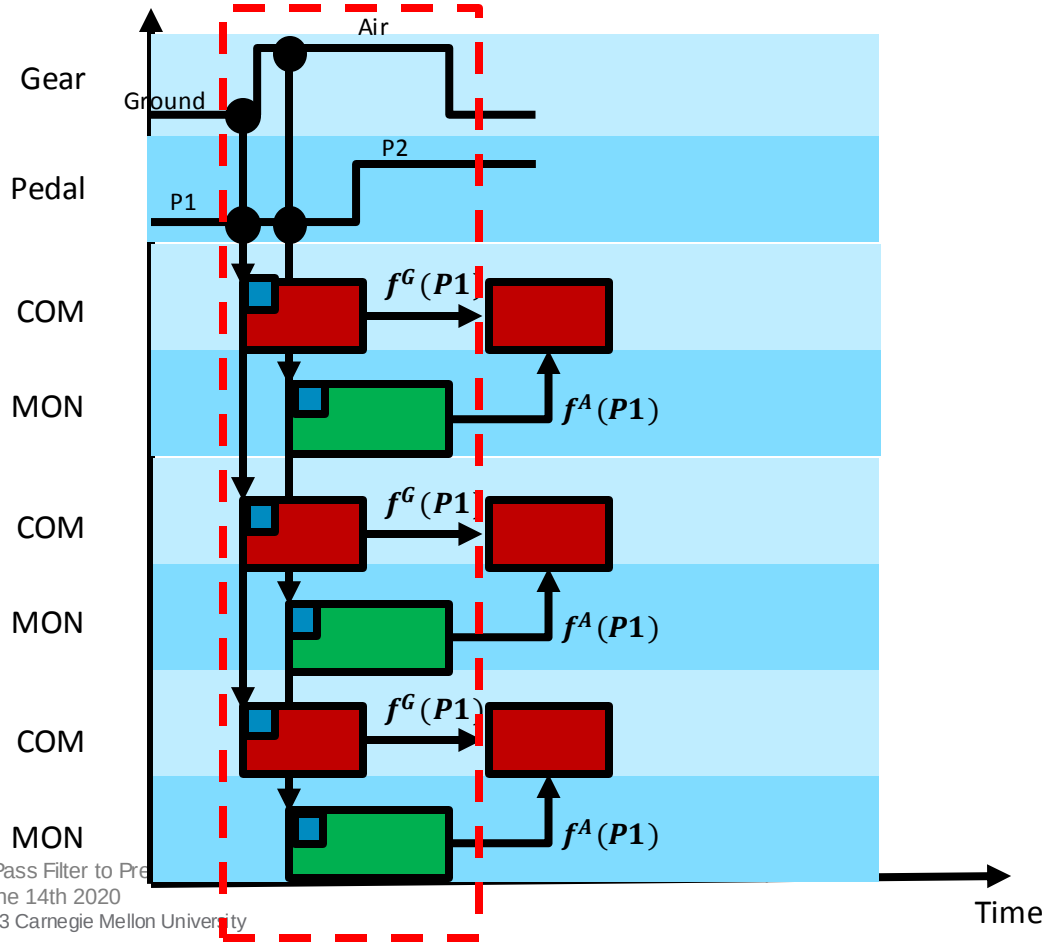



```
Initialization
            mrt := 0; mtf := 0; count := 0
When the replica gets a new sensor reading from the ground
sensor (approximately periodically), do the following
  Let s denote the value of the ground sensor
  count := count + 1
  if s is True then
      mrt := mrt + 1
      mrf := 0
    else
      mrf := mrf + 1
      mrt := 0
  if count<WL then
    output <f,False> where f is arbitrary
  else
    if mrt>=WL then
      last_confident_output := True
      output <True,True>
    else
      if mrf>=WL then
        last_confident_output := False
        output <False,True>
      else
        output <last_confident_output,False>
```

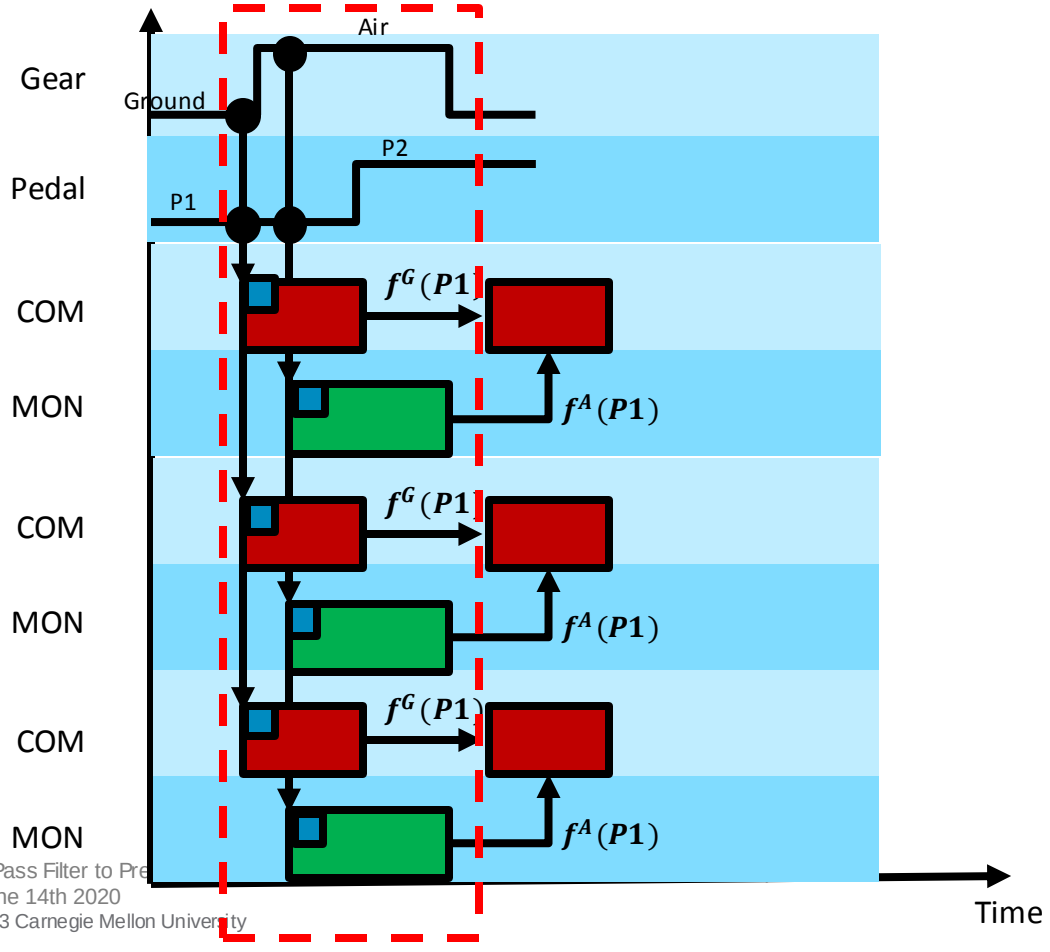
How do we deal with non-periodic sampling.



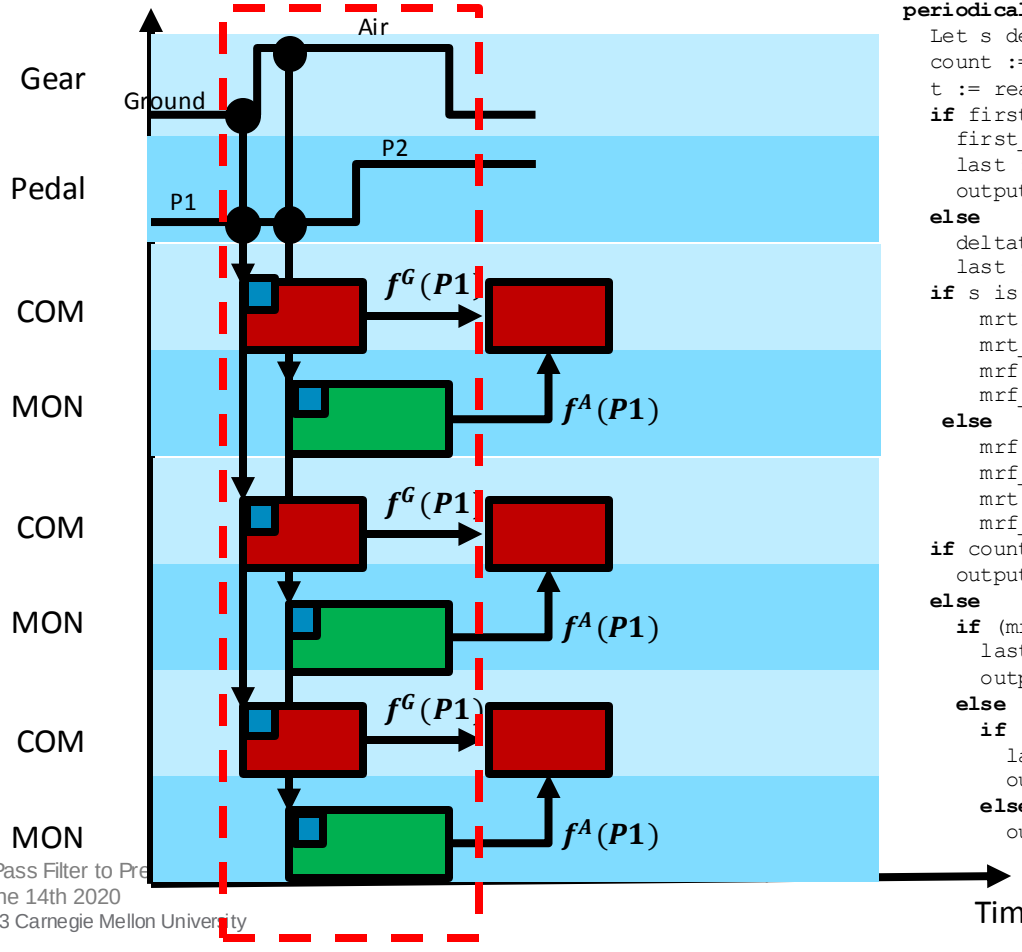
In this case, if we get WL sensor inputs in very short duration, and they are all True, then the filter should not output True (because there may be correlated noise of all the WL sensor inputs).



We should require that we get WL sensor inputs that are the same and they should span a sufficiently large duration of time; only then should the filter produce output with high confidence.



We should require that we get WL sensor inputs that are the same and they should span a sufficiently large duration of time; only then should the filter produce output with high confidence.




```
Initialization
     mrt := 0; mtf := 0; count := 0;
     mrt_time := 0.0; mrf_time := 0.0;
     first_sensor_reading := True
When the replica gets a new sensor reading from the ground sensor (approximately
periodically), do the following
  Let s denote the value of the ground sensor
  count := count + 1
  t := read_current_time (or let t be timestamp from ground sensor)
  if first_sensor_reading then
    first_sensor_reading := False
    last := t
    output <b,False> where b can take any value (False or True)
  else
    deltat := t-last
    last := t
  if s is True then
      mrt      := mrt + 1
      mrt_time := mrt_time + delta
      mrf      := 0
      mrf_time := 0.0
   else
      mrf      := mrf + 1
      mrf_time := mrf_time + delta
      mrt      := 0
      mrf_time := 0.0
  if count<WL then
    output <f,False> where f is arbitrary
  else
    if (mrt>=WL) and (mrt_time>=WLTIME) then
      last_confident_output := True
      output <True,True>
    else
      if (mrf>=WL) and (mrf_time>=WLTIME) then
        last_confident_output := False
        output <False,True>
      else
        output <last_confident_output,False>
```