



**Rolls-Royce**

# **Decentralized Engine Control System Simulator DECSS**

The hard real-time platform with  
analog and digital I/O for hardware testing

John McArthur, Bobbie Hegwood, O.A. (Bud) Watts

11 December 2013

© 2013 Rolls-Royce Corporation

# Agenda

- Why DECSS and HIL
- DECSS Description
  - Hardware
  - Capabilities
- DECSS Models and Suggested Architecture
- HIL Testing – DECSS Efficiencies
  - Legacy
  - Distributed
- DECSS Operation
  - | Flexible and Simplified Modeling from all simulation to HIL
- Summary

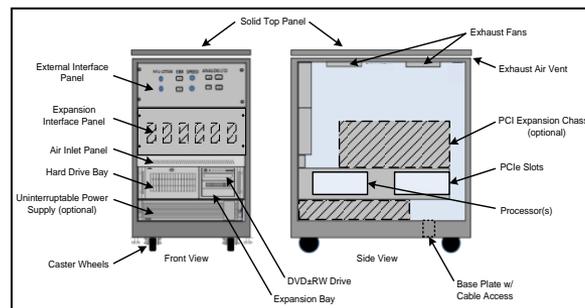
# HIL testing is important

## DECSS provides the environment

- High fidelity testing of hardware in the loop provides validation of all the interfaces and logic before risking expensive hardware and life on the test stand.
  - Facilitates rapid and inexpensive debugging of hardware components
  - Enables low-risk and low-cost environment iterative development with real control system hardware
  - Allows identification and verification of realistic minimum bandwidth, latencies, packet drop probabilities, and redundancies
  - Facilitates analysis of sensitivities to off-nominal component performance
    - Off-design engine, controller, actuator, sensor, and network can be tested for (system and/or component) robustness with low-cost and low-risk

# DECSS Description

- DECSS is a hard Real-Time capable Simulation and Test Environment developed to support Distributed and Decentralized Engine Control development and integration with vehicles
  - | Hardware is Linux based real-time operating system with interfaces compatible with WinOS and multiple control system communication bus structures plus 16 channels of A/D and 4 channels of D/A
  - | Software is built around industry standard MatLab Simulink with support packages to interface with multiple external platforms and to simplify Hardware In the Loop testing and evaluation of developmental systems

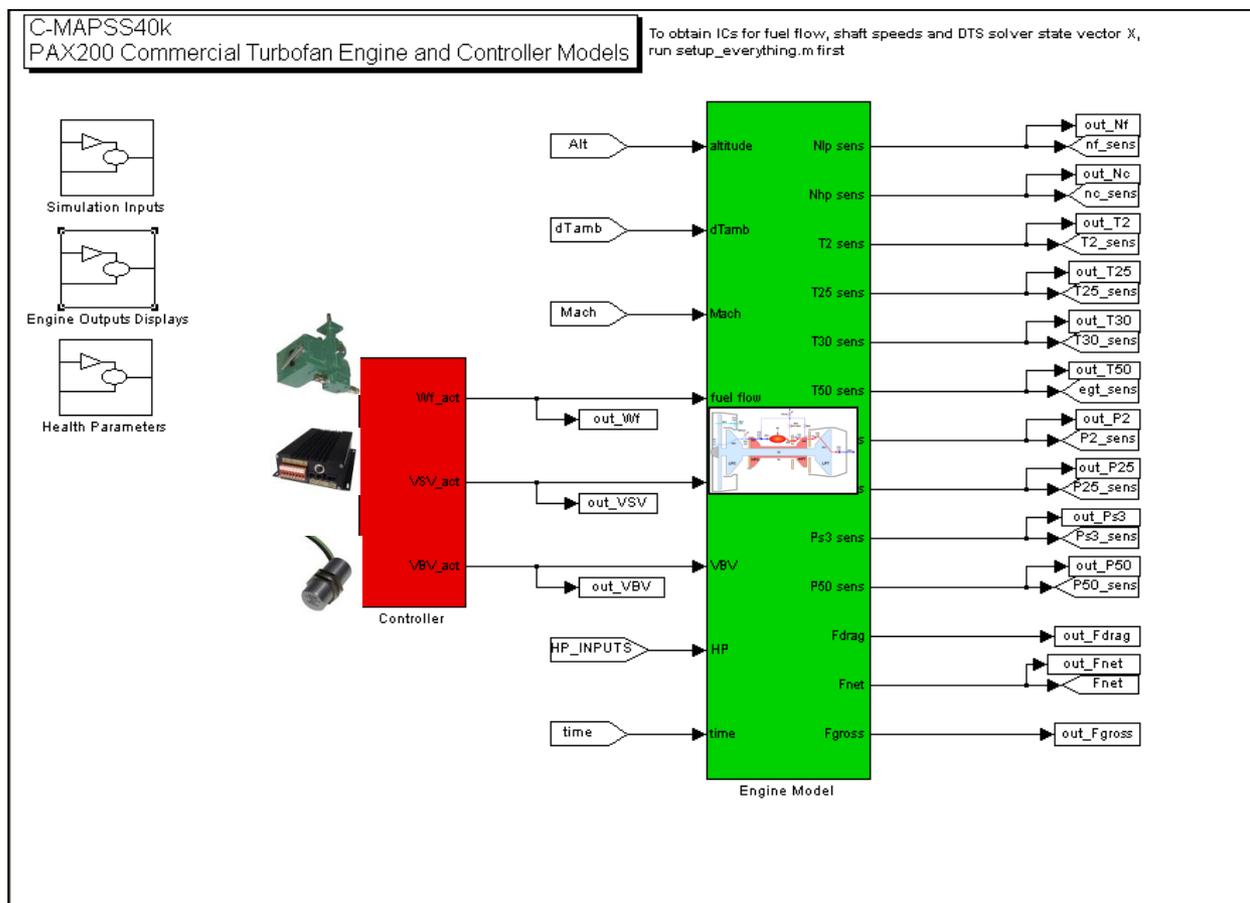


# Models and Architecture

C-MAPSS40K  
Original

## Limitations:

- Single .mdl file
  - Replacing software with hardware is more difficult
  - Parallel development of model is more difficult
- Use of to/from blocks
  - Can introduce trouble in code generation
  - Cannot see system dependencies

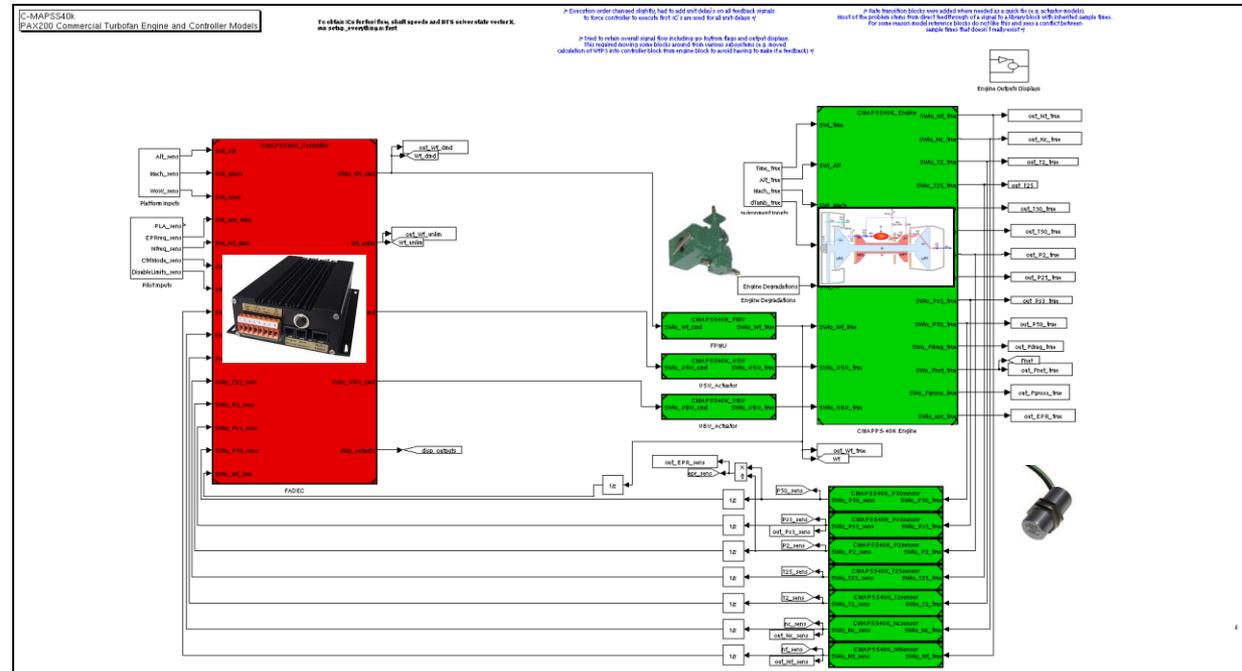


# Models and Architecture targeted for DECSS

C-MAPSS40K  
Model Referenced

## Improvements:

- Blocks are .mdl files
  - Creates individual executables that can be replaced by hardware
  - Developers can work without worry of conflicts with other developers
- No to/from blocks
  - Eliminates concern of improper code generation
  - System dependencies are more clear



# DECSS Models

## Engine with Lumped Network Model

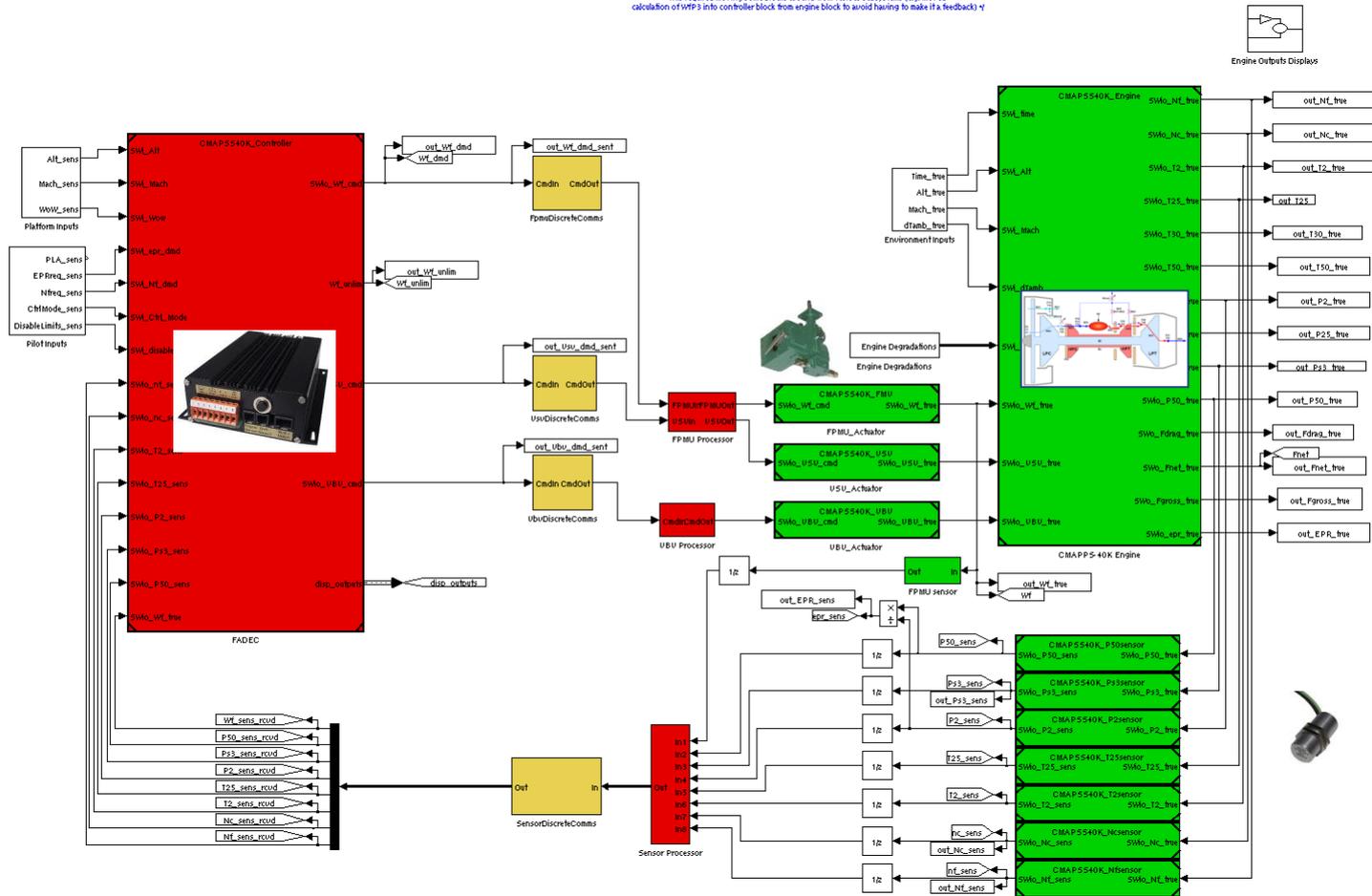
C-MAPSS40K  
PAX200 Commercial Turbofan Engine and Controller Models

To obtain ICs for fuel flow, shaft speeds and DTS solver state vector X, run setup\_everything.m first

\* Execution order changed slightly, had to add unit delay's on all feedback signals to force controller to execute first IC's are used for all unit delay's \*

\* Tried to retain overall signal flow including go-to-from flags and output displays. This required moving some blocks around from various subroutines (e.g. moved calculation of WFP3 into controller block from engine block to avoid having to make it a feedback \*)

\* Rate transition blocks were added where needed as a quick fix (e.g. actuator models). Most of the problem stems from direct feedthrough of a signal to a library block with inherited sample times. For some reason model reference blocks do not like this and sees a conflict between sample times that doesn't really exist! \*



# DECSS Models

## Engine with Lumped Network Model

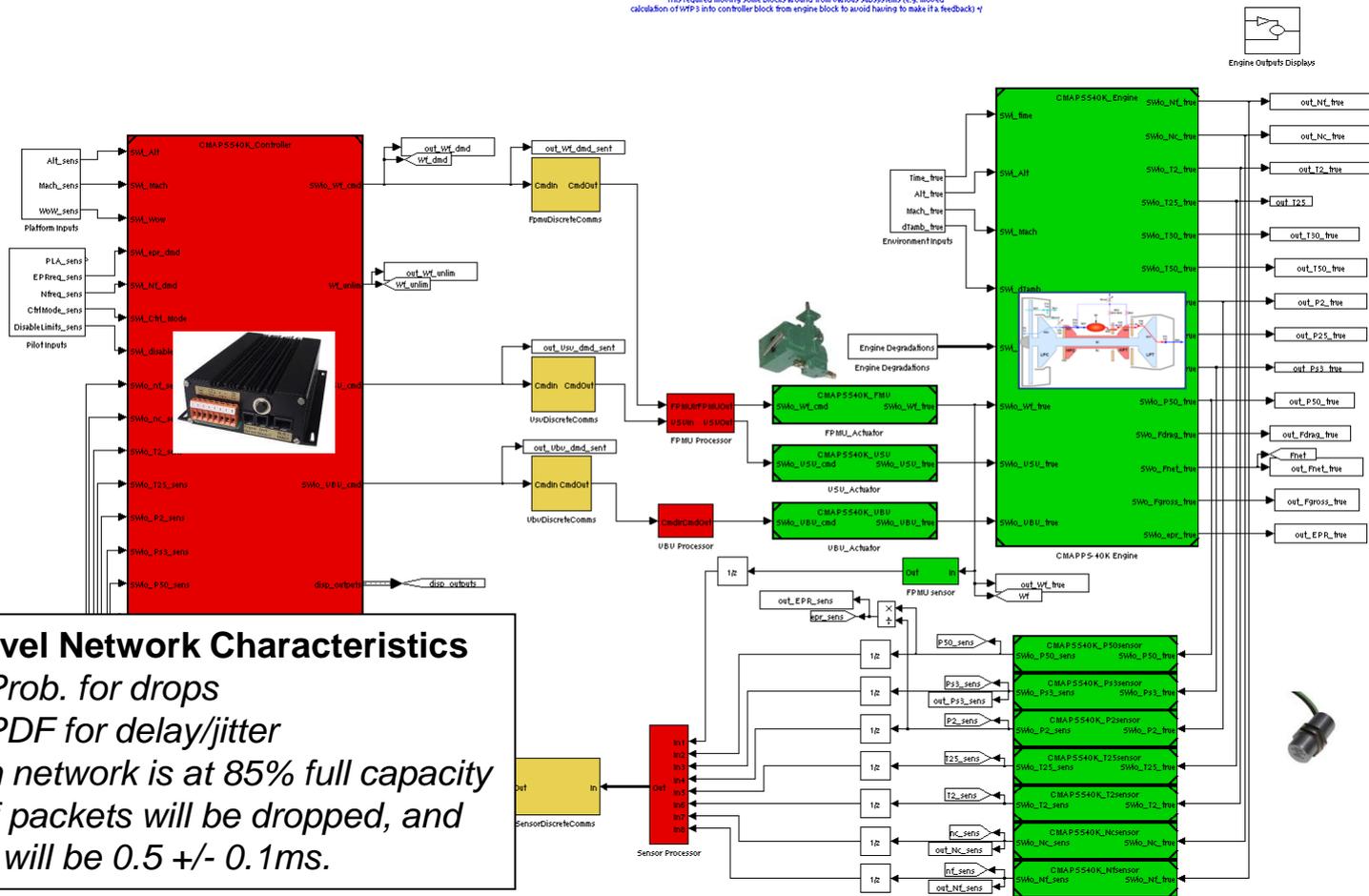
C-MAPSS40K  
PAX200 Commercial Turbofan Engine and Controller Models

To obtain ICs for fuel flow, shaft speeds and DTs solver state vector X, run `setbu_everything.m` first

\* Execution order changed slightly, had to add unit delays on all feedback signals to force controller to execute first IC's are used for all unit delays \*

\* Tried to retain overall signal flow including go-faults flags and output displays. This required moving some blocks around from various subsystems (e.g. moved calculation of WPP3 into controller block from engine block to avoid having to make it a feedback) \*

\* Rate transition blocks were added where needed as a quick fix (e.g. actuator models). Most of the problem stems from direct feedthrough of a signal by a library block with inherited sample times. For some reason model reference blocks do not like this and sees a conflict between sample times that doesn't really exist \*



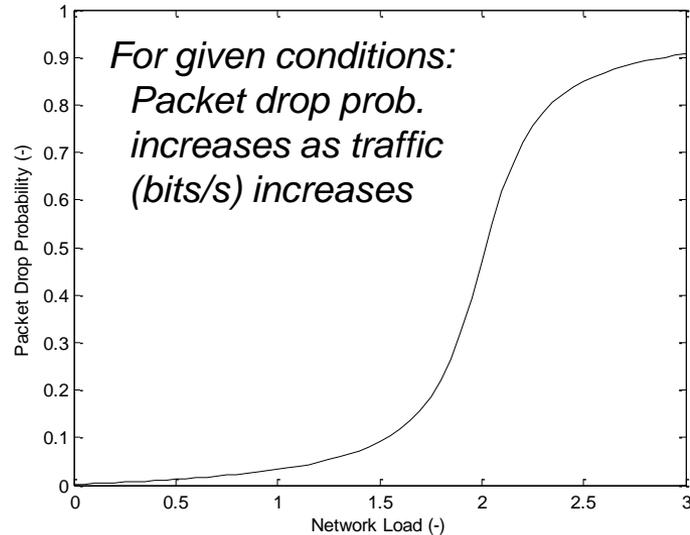
**High- Level Network Characteristics**

- Use Prob. for drops
- Use PDF for delay/jitter
- When network is at 85% full capacity 5% of packets will be dropped, and delay will be 0.5 +/- 0.1ms.

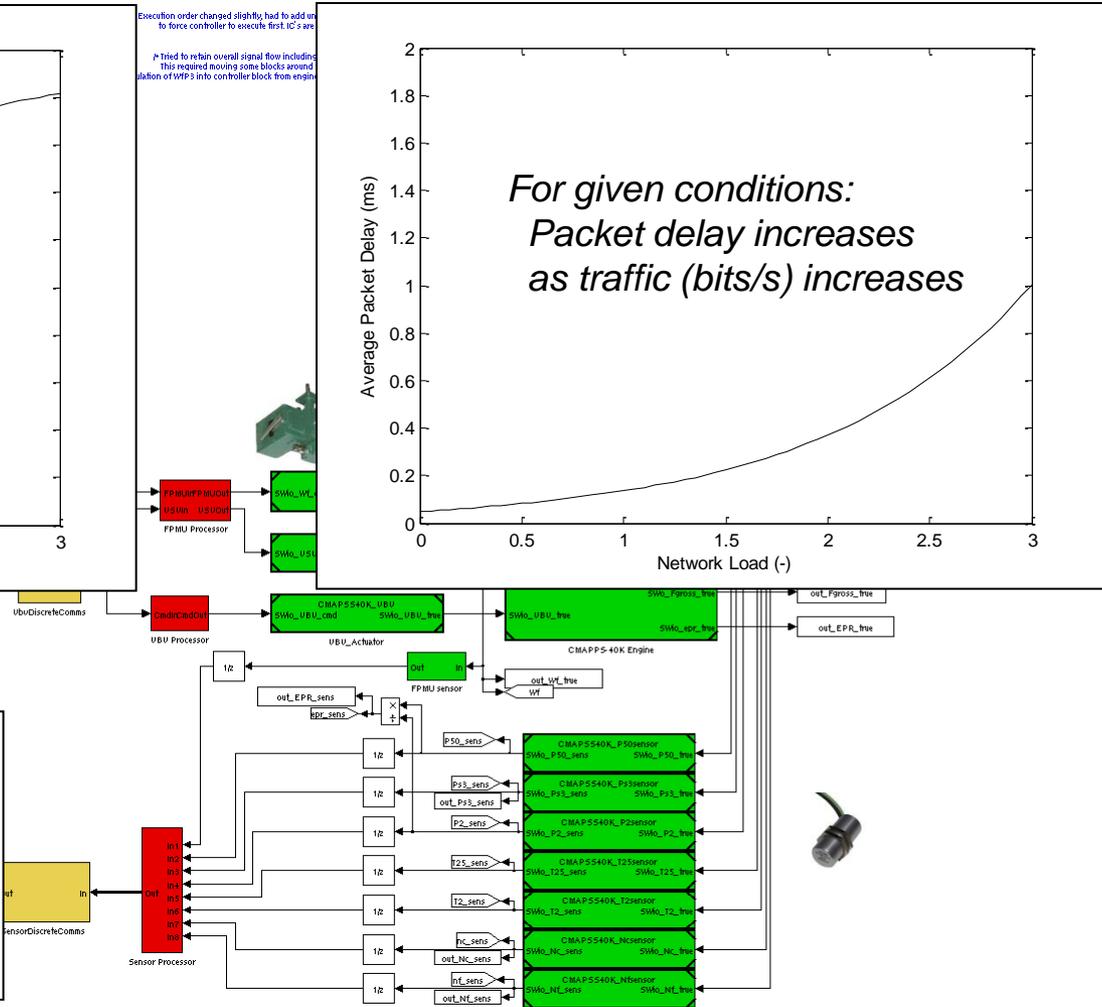
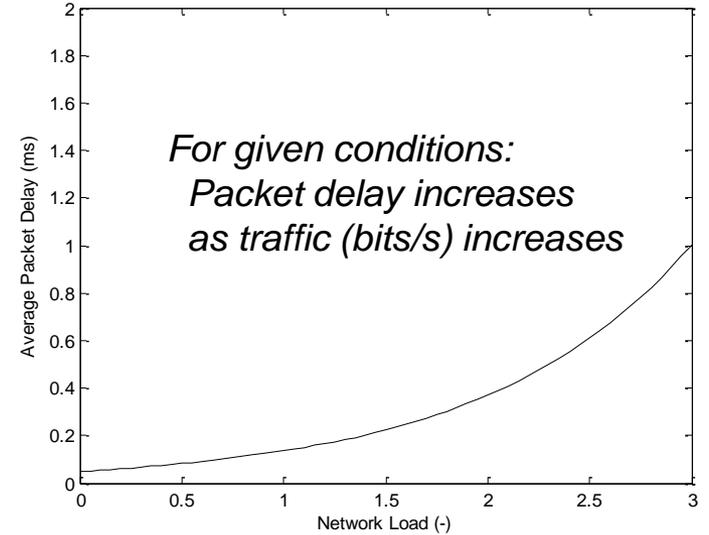
# DECSS Models

Engine with Lumped Network Model

C-MAP  
PAX20



Execution order changed slightly, had to add in  
to force controller to execute first IC's etc  
  
P tried to retain overall signal flow including  
This required moving some blocks around  
station of VVP's into controller block from engine



## High- Level Network Characteristics

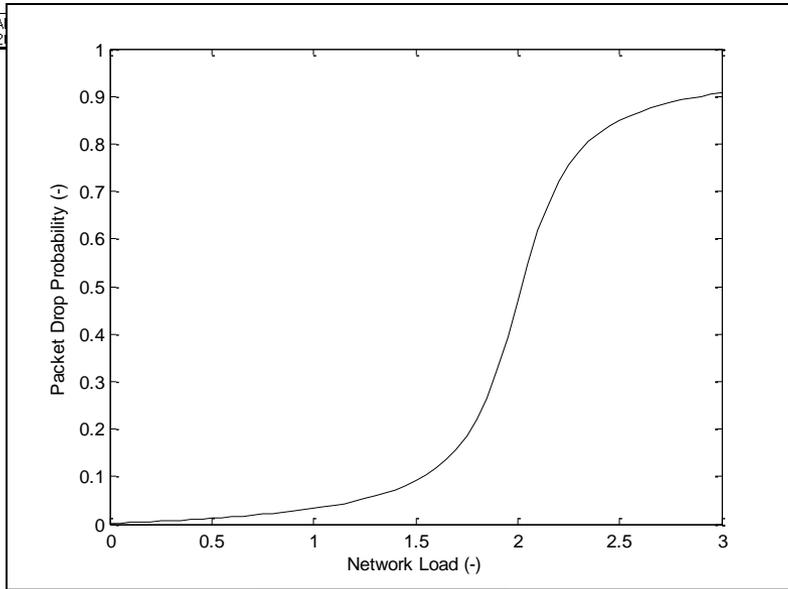
- Use Prob. for drops
- Use PDF for delay/jitter
- When network is at 85% full capacity 5% of packets will be dropped, and delay will be 0.5 +/- 0.1ms.



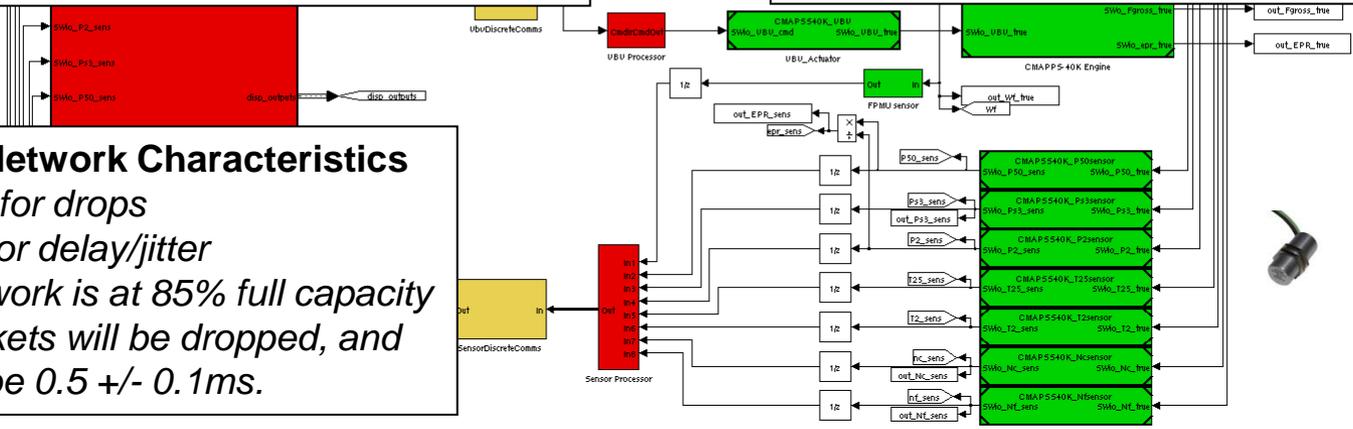
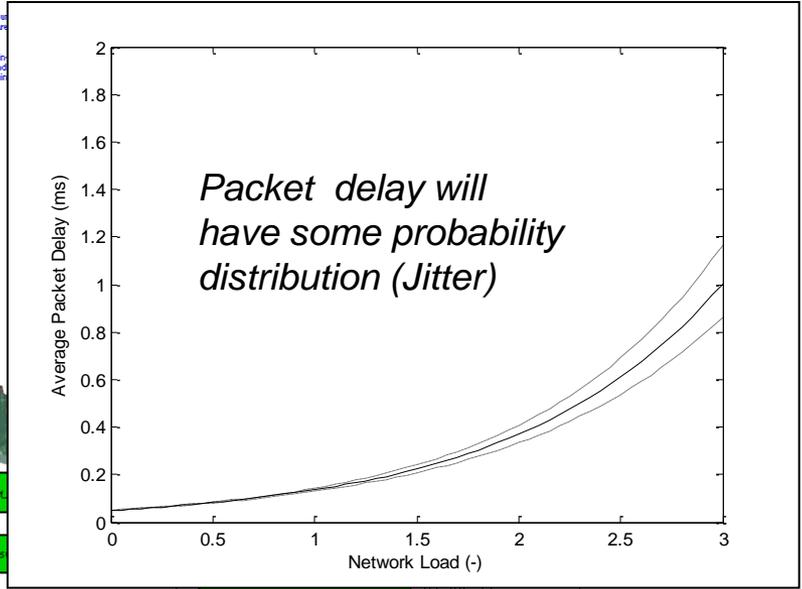
# DECSS Models

Engine with Lumped Network Model

CMA PAX2



Execution order changed slightly, had to add ...  
to force controller to execute first IC's ...  
  
I tried to retain overall signal flow including ...  
This required moving some blocks around ...  
relaxation of WPP3 into controller block from engine



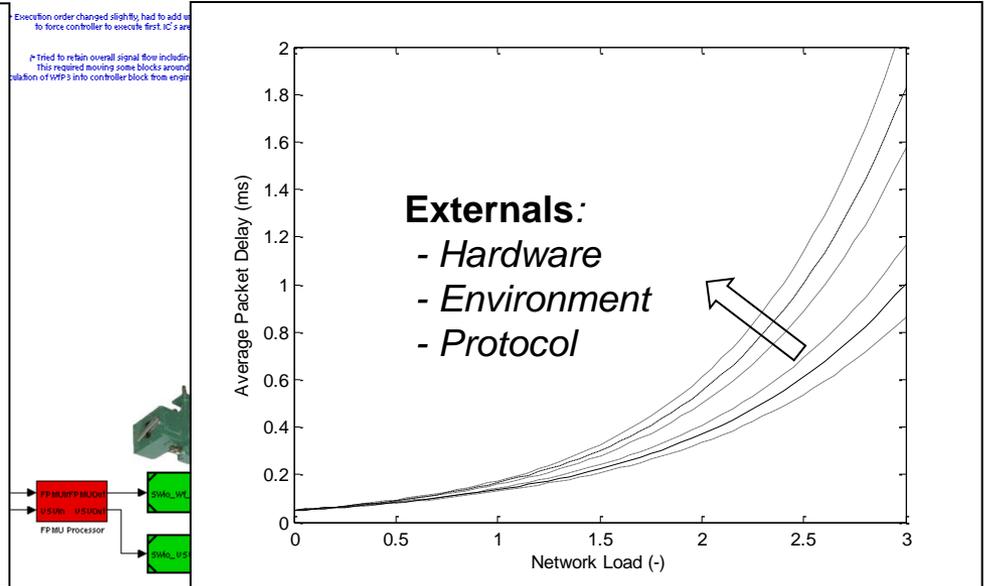
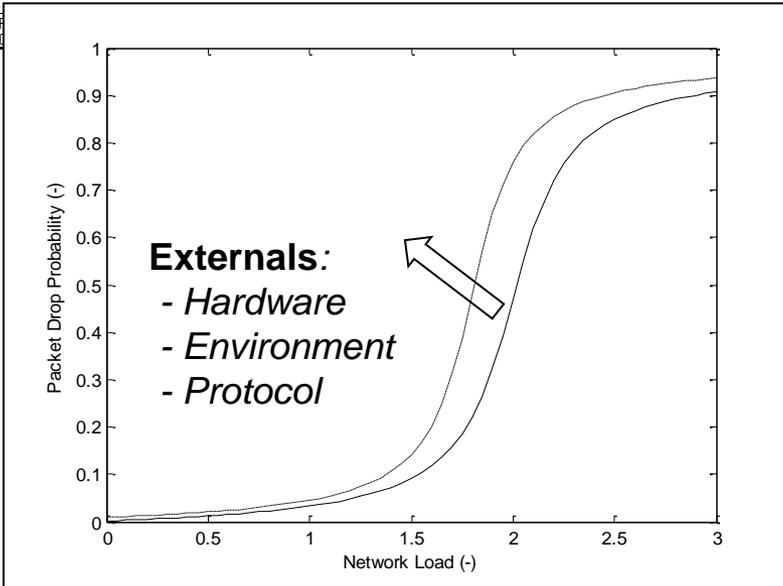
**High- Level Network Characteristics**

- Use Prob. for drops
- Use PDF for delay/jitter
- When network is at 85% full capacity 5% of packets will be dropped, and delay will be 0.5 +/- 0.1ms.

# DECSS Models

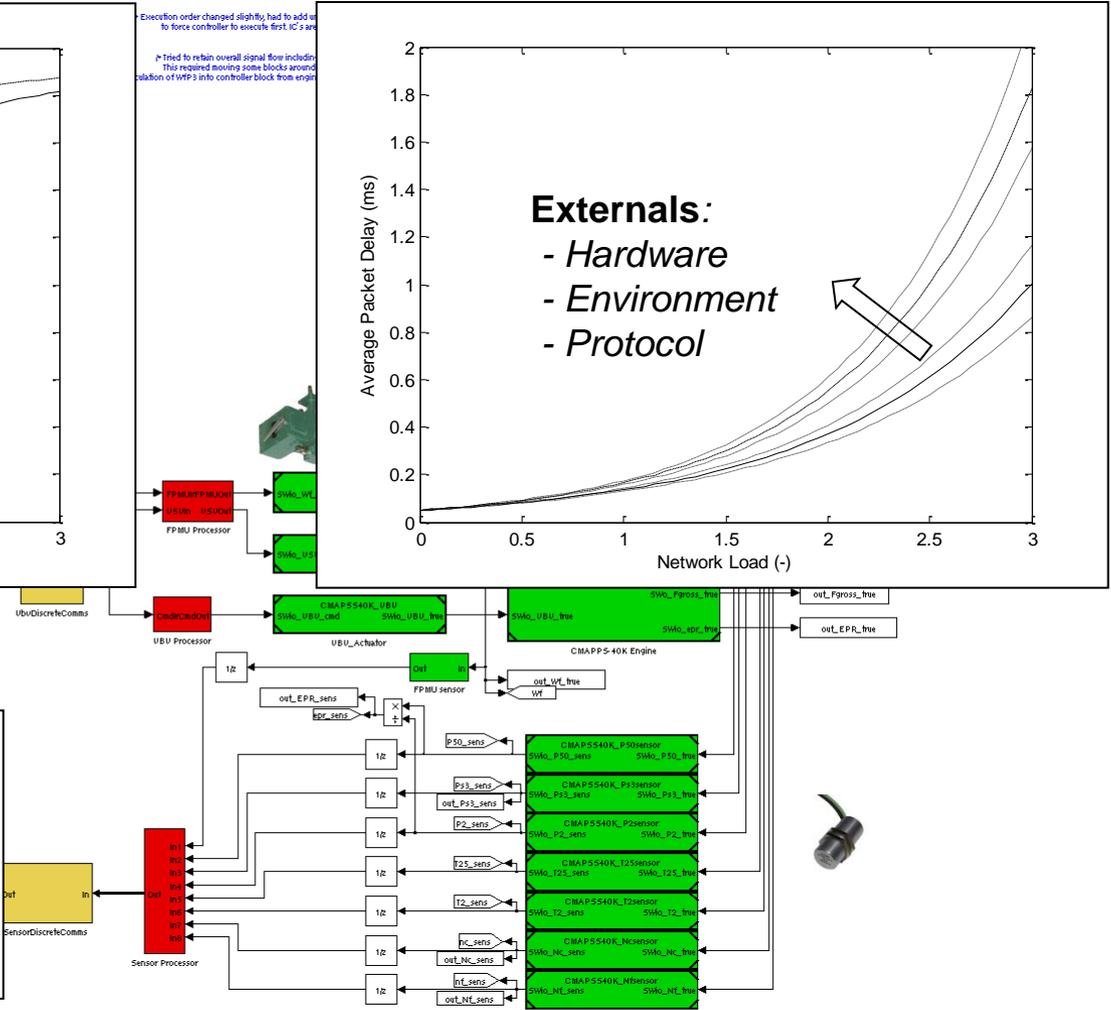
Engine with Lumped Network Model

C-MAP PAX20



**High- Level Network Characteristics**

- Use Prob. for drops
- Use PDF for delay/jitter
- When network is at 85% full capacity 5% of packets will be dropped, and delay will be 0.5 +/- 0.1ms.



# DECSS Models

## Simulation with Detailed Network Model

C-MAPSS40K  
PAX200 Commercial Turbofan Engine and Controller Models

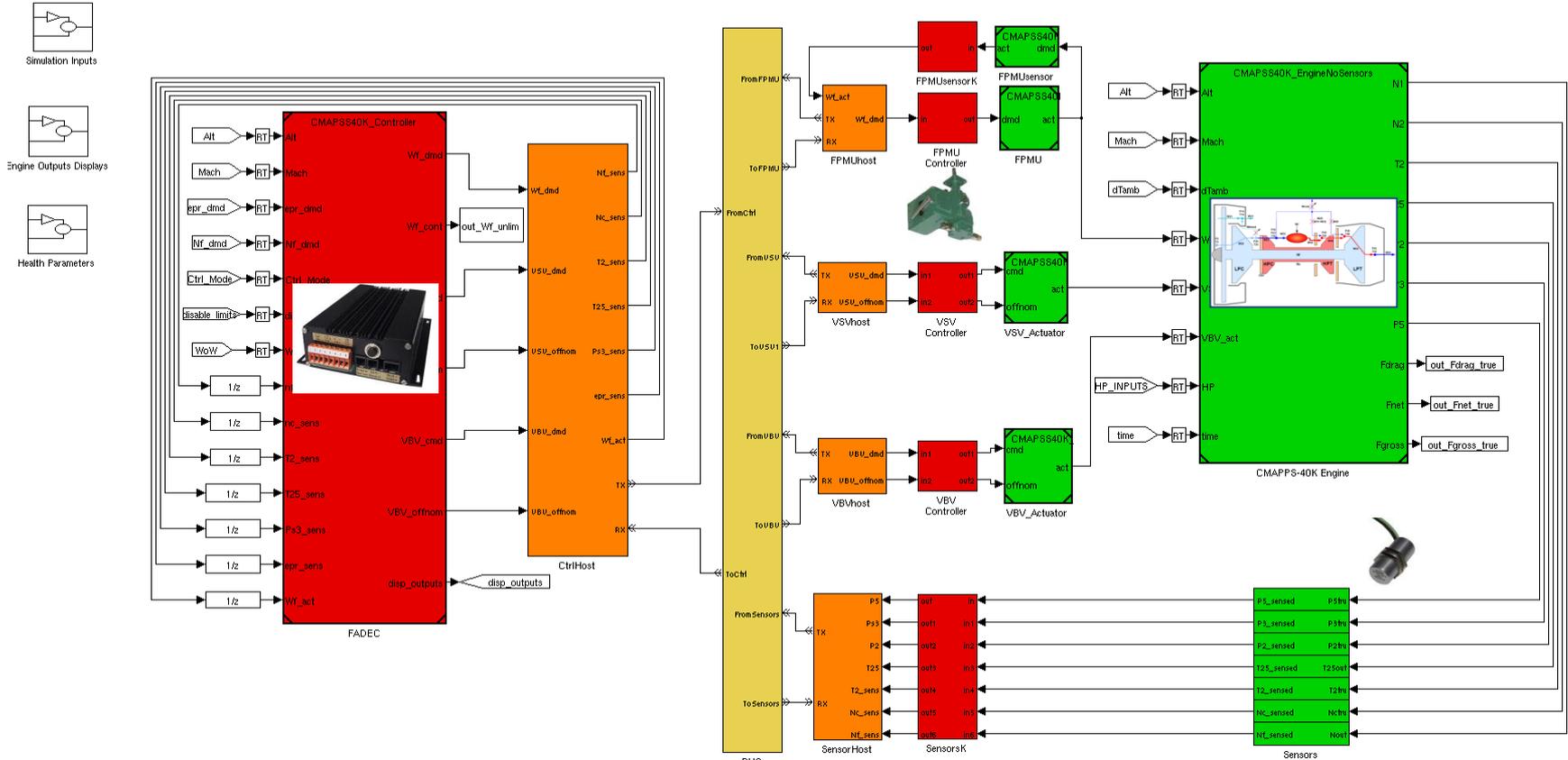
To obtain ICs for fuel flow, shaft speeds and DTS solver state vector X, run setup\_everything.m first

/\* Execution order changed slightly, had to add unit delay's on all feedback signals to force controller to execute first. ICs are used for all unit-delays.

Also, with SimEvents, lack of unit delays caused algebraic loops in the discrete model that could not be solved by Simulink. Using an IC block did not fix the problem, must be unit delay \*/

/\* Rate transition blocks were added where needed as a quick fix (e.g. actuator models). Most of the problem stems from direct feedthrough of a signal to a library block with inherited sample times. For some reason model reference blocks do not like this and sees a conflict between sample times that doesn't really exist \*/

/\* Tried to retain overall signal flow including go-to/from flags and output displays. This required moving some blocks around from various subsystems (e.g. moved calculation of WFP3 into controller block from engine block to avoid having to make it a feedback) \*/



# DECSS Models

## Simulation with Detailed Network Model

C-MAPSS40K  
PAX200 Commercial Turbofan Engine and Controller Models

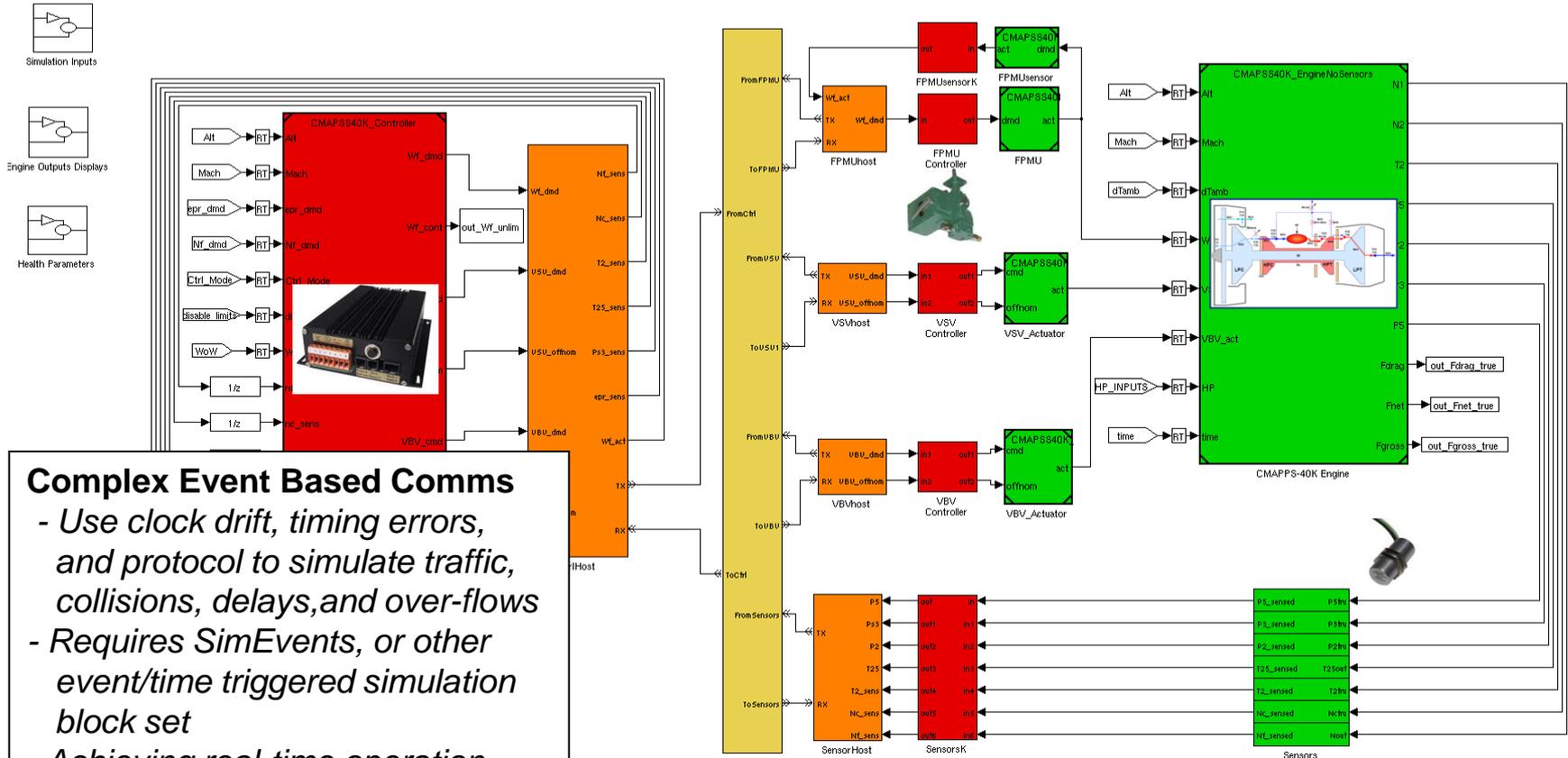
To obtain ICs for fuel flow, shaft speeds and DTS solver state vector X, run setup\_everything\_in first

/\* Execution order changed slightly, had to add unit delay's on all feedback signals to force controller to execute first. ICs are used for all unit-delays.

Also, with SimEvents, lack of unit delays caused algebraic loops in the discrete model that could not be solved by Simulink. Using an IC block did not fix the problem, must be unit delay \*/

/\* Rate transition blocks were added where needed as a quick fix (e.g. actuator models). Most of the problem stems from direct feedthrough of a signal to a library block with inherited sample times. For some reason model reference blocks do not like this and sees a conflict between sample times that doesn't really exist \*/

/\* Tried to retain overall signal flow including go-to/from flags and output displays. This required moving some blocks around from various subsystems (e.g. moved calculation of WFP3 into controller block from engine block to avoid having to make it a feedback) \*/



**Complex Event Based Comms**

- Use clock drift, timing errors, and protocol to simulate traffic, collisions, delays, and over-flows
- Requires SimEvents, or other event/time triggered simulation block set
- Achieving real-time operation can be difficult

# DECSS Models

## Simulation with Detailed Network Model

C-MAPSS40K  
PAX200 Commercial Turbofan Engine and Controller Models

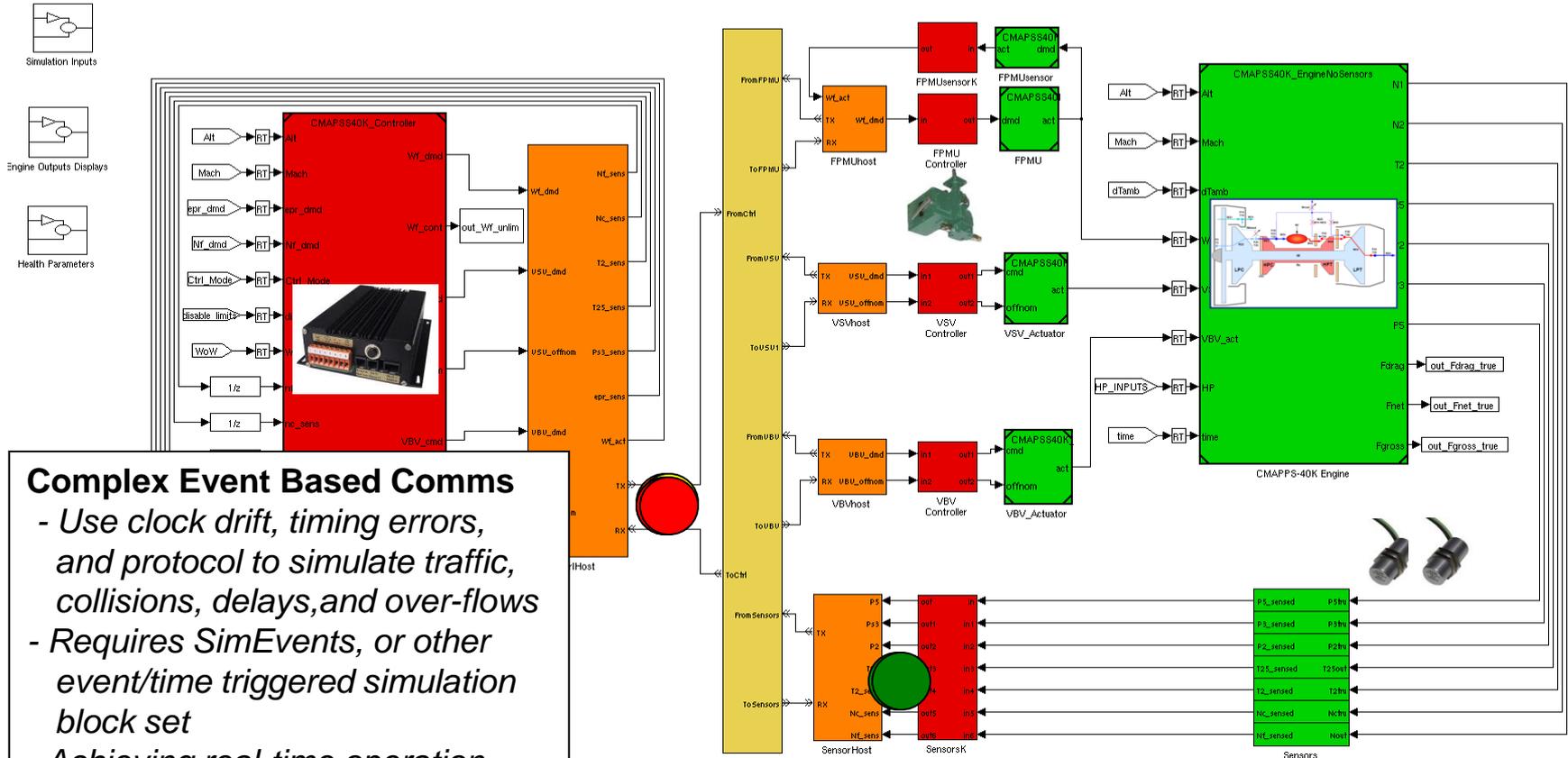
To obtain ICs for fuel flow, shaft speeds and DTS solver state vector X, run setup\_everything in first

/\* Execution order changed slightly, had to add unit delay's on all feedback signals to force controller to execute first. ICs are used for all unit-delays.

Also, with SimEvents, lack of unit delays caused algebraic loops in the discrete model that could not be solved by Simulink. Using an IC block did not fix the problem, must be unit delay \*/

/\* Rate transition blocks were added where needed as a quick fix (e.g. actuator models). Most of the problem stems from direct feedthrough of a signal to a library block with inherited sample times. For some reason model reference blocks do not like this and sees a conflict between sample times that doesn't really exist \*/

/\* Tried to retain overall signal flow including go-to/from flags and output displays. This required moving some blocks around from various subsystems (e.g. moved calculation of WFP3 into controller block from engine block to avoid having to make it a feedback) \*/

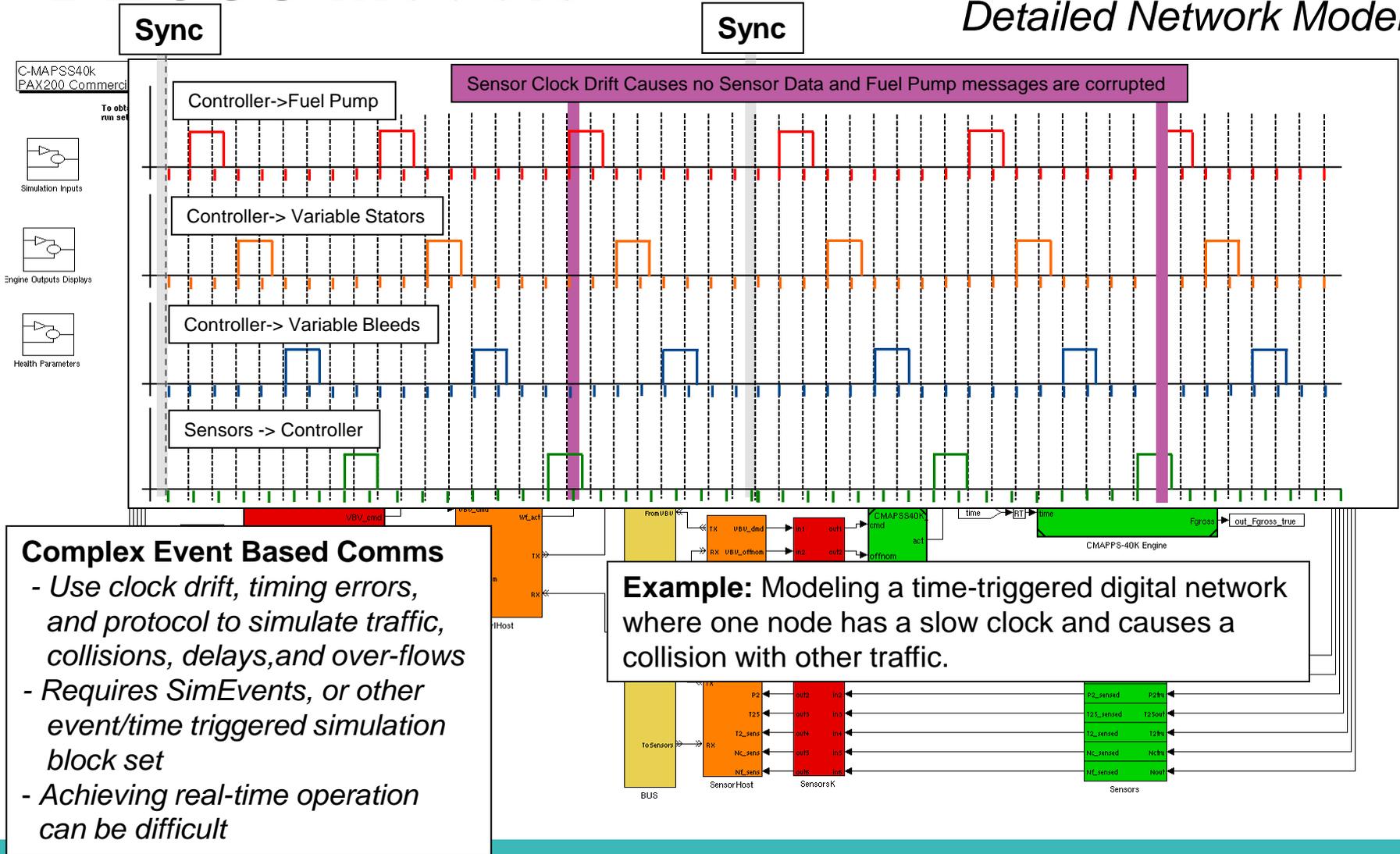


**Complex Event Based Comms**

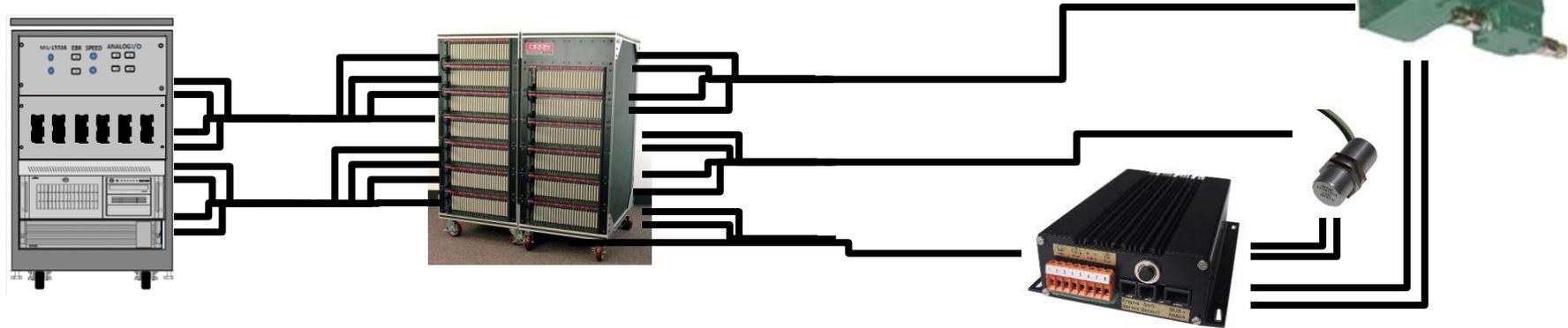
- Use clock drift, timing errors, and protocol to simulate traffic, collisions, delays, and over-flows
- Requires SimEvents, or other event/time triggered simulation block set
- Achieving real-time operation can be difficult

# DECSS Models

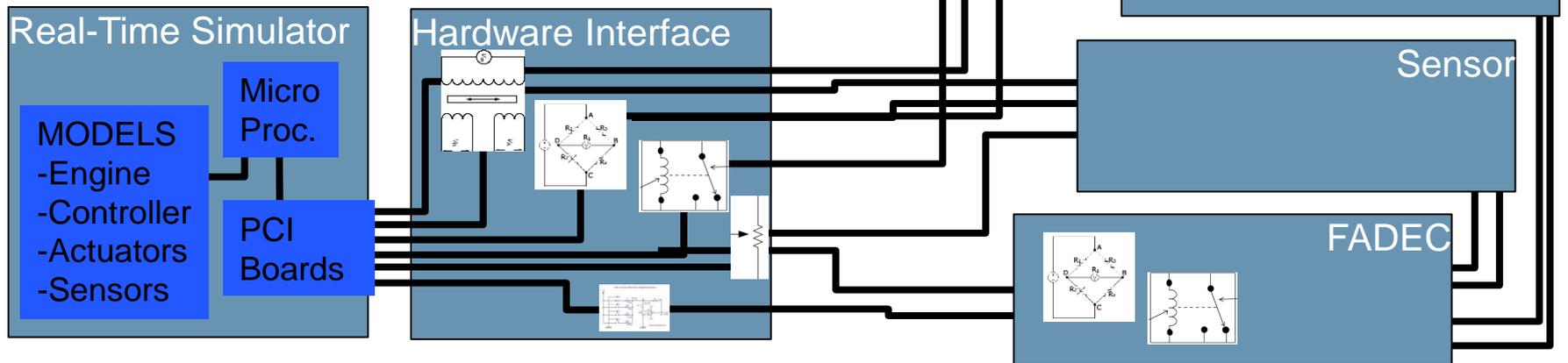
*Simulation with Detailed Network Model*



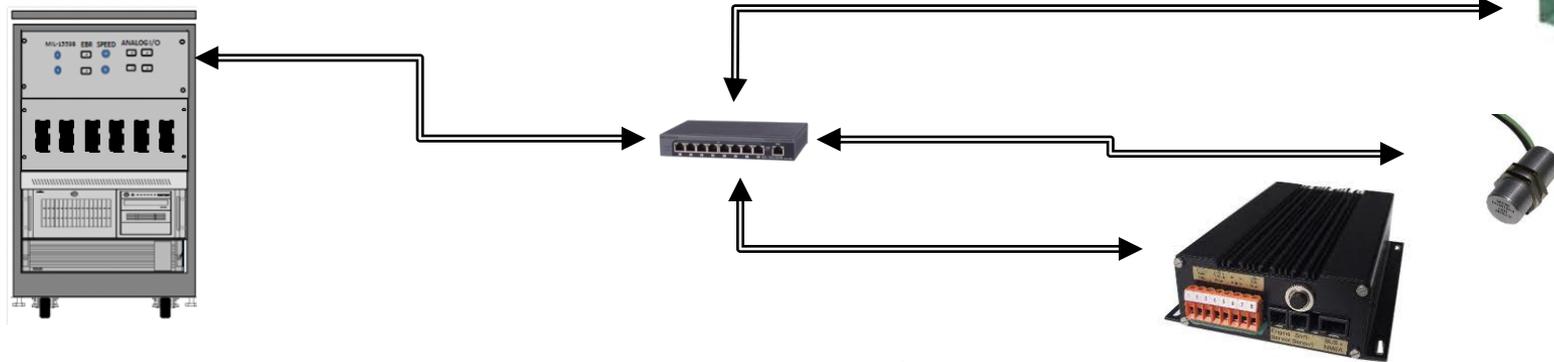
# Component Testing - Legacy



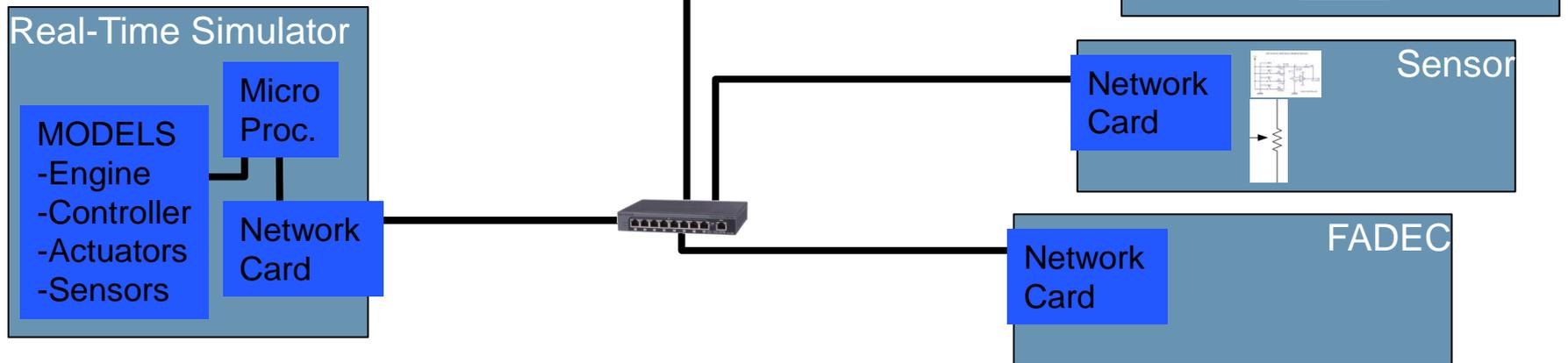
Legacy systems have multiple complex analog interfaces specific to each component/vendor/design/implementation



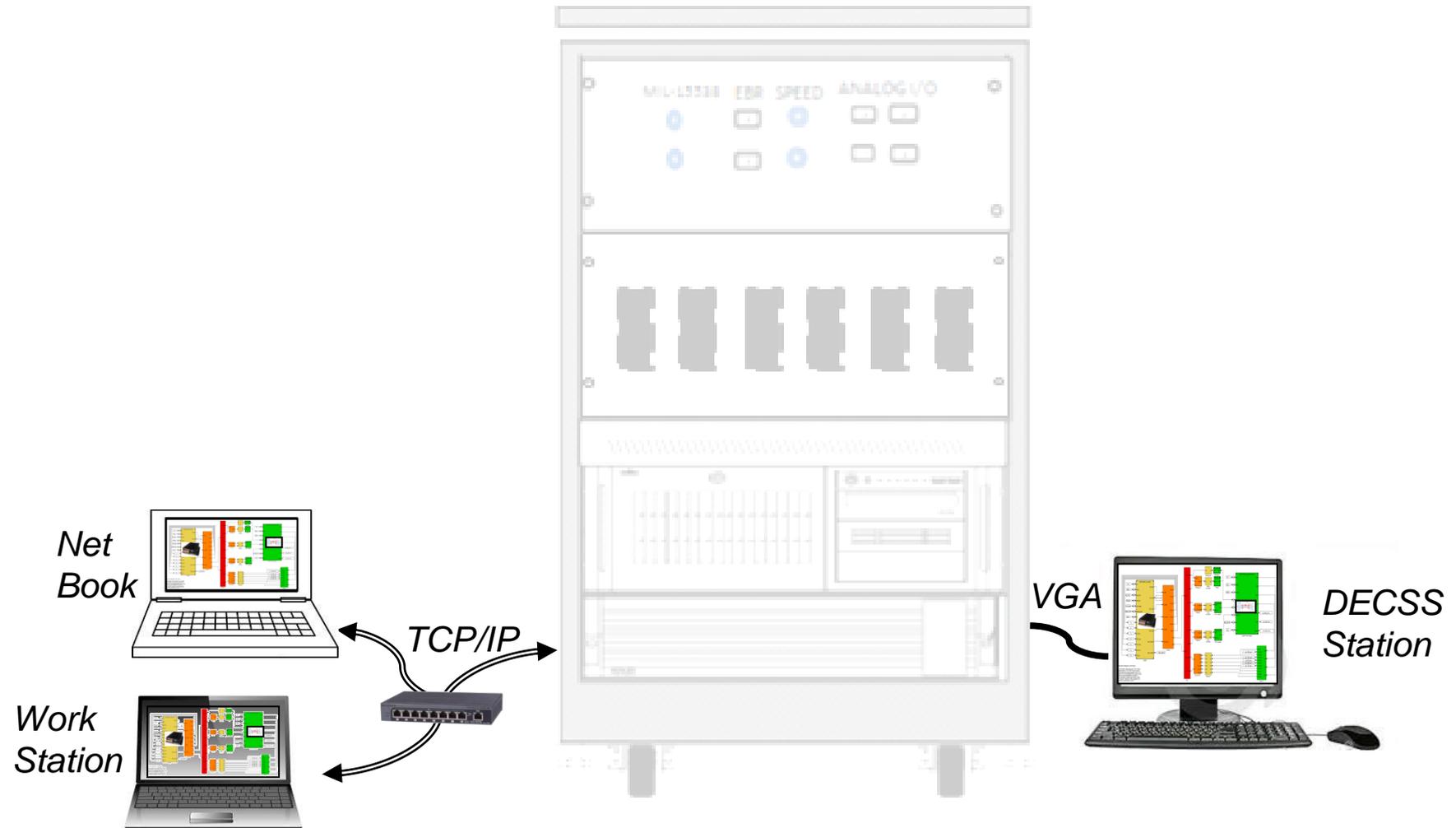
# Component Testing - Distributed



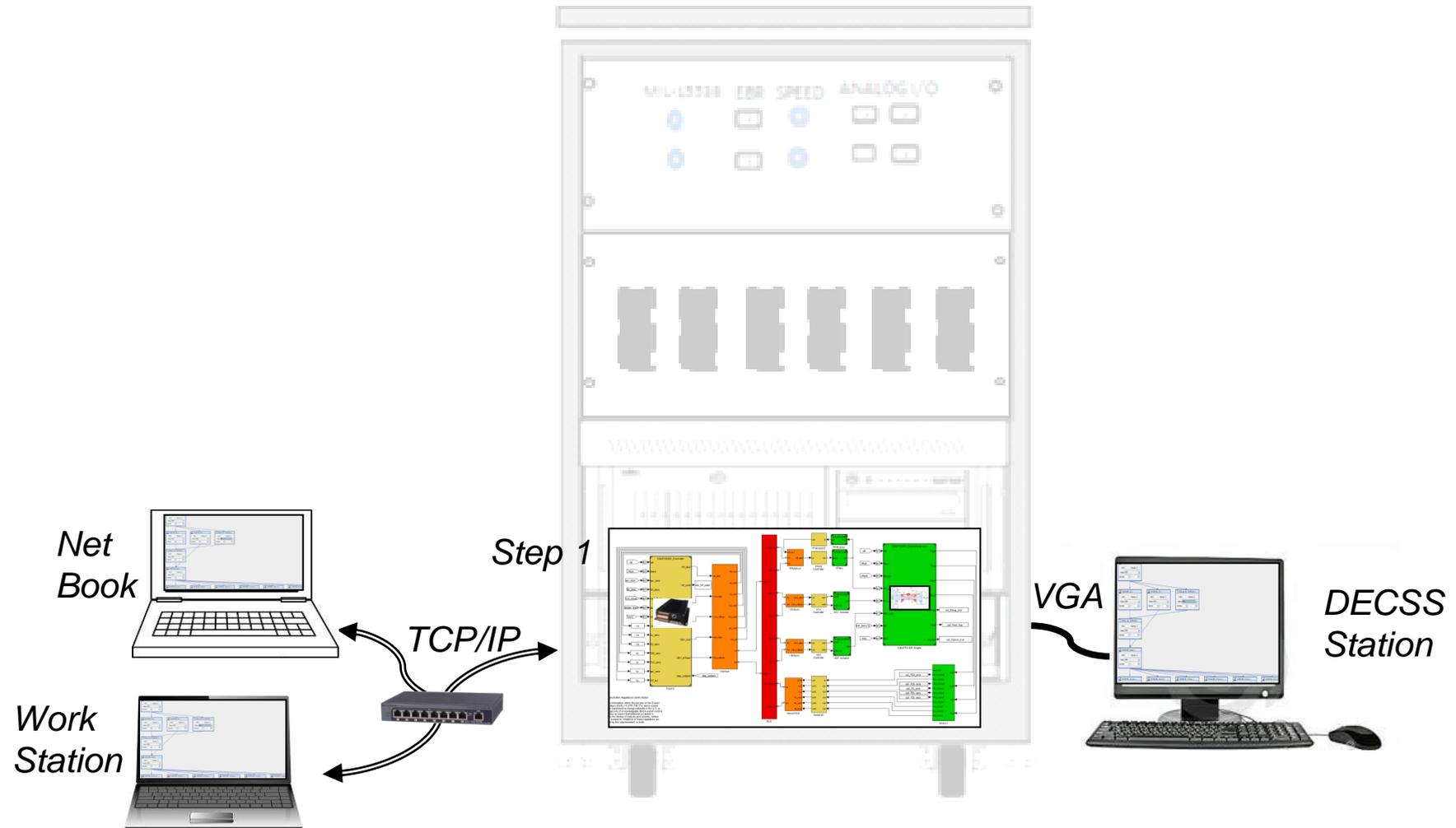
Distributed systems have a single simplified open digital interface with analog elements contained in the components



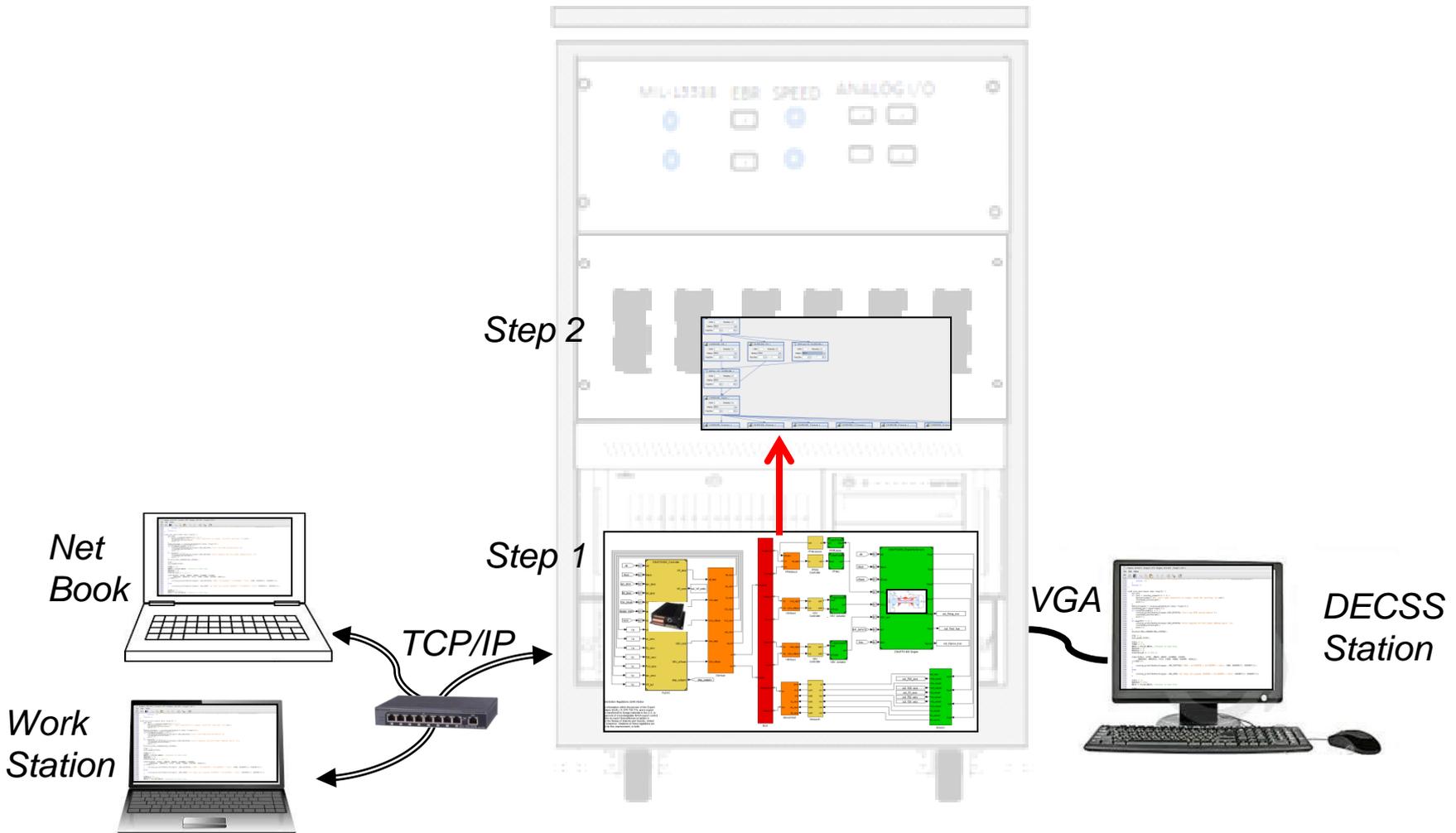
# DECSS Operations



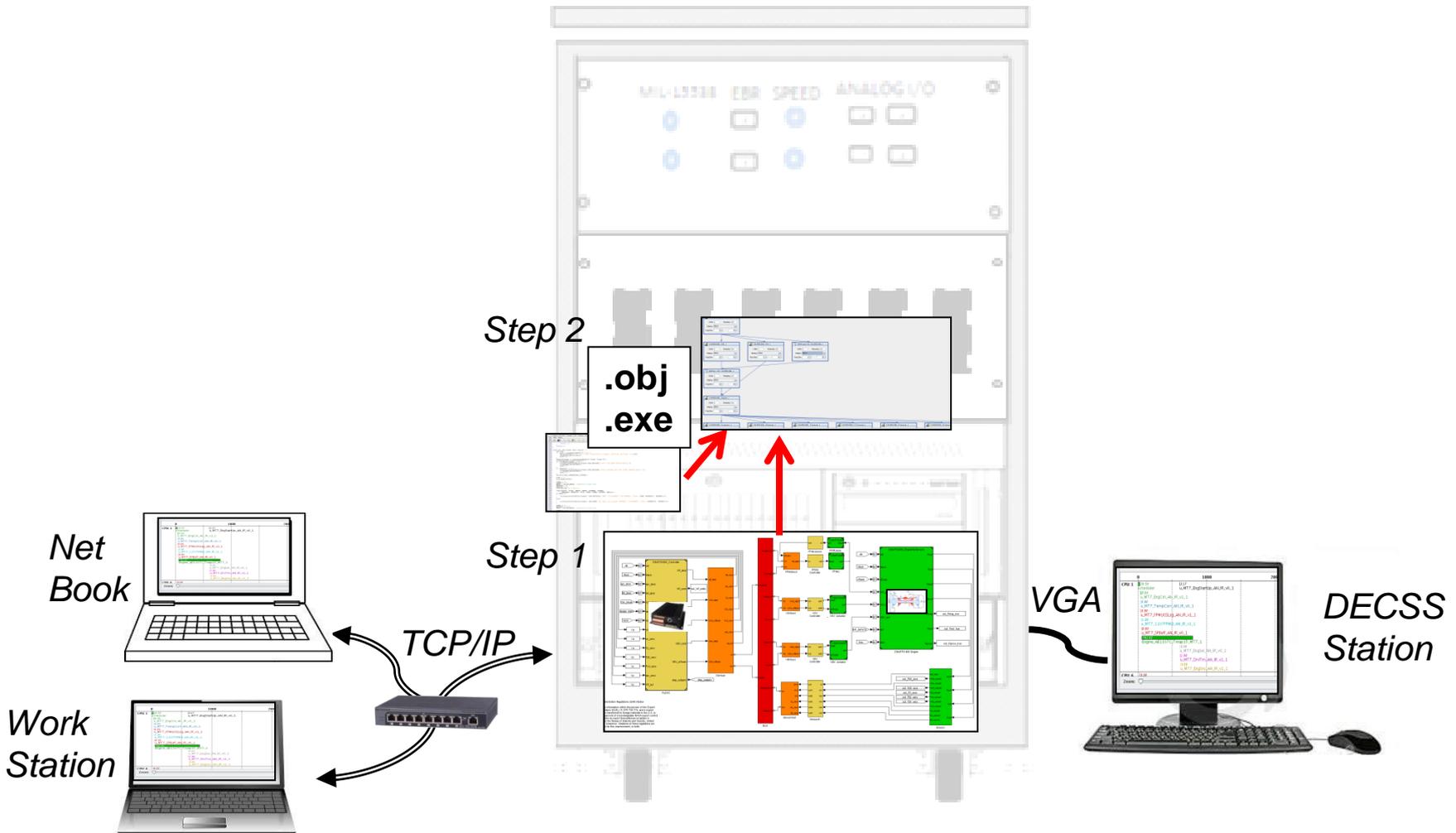
# DECSS Operations



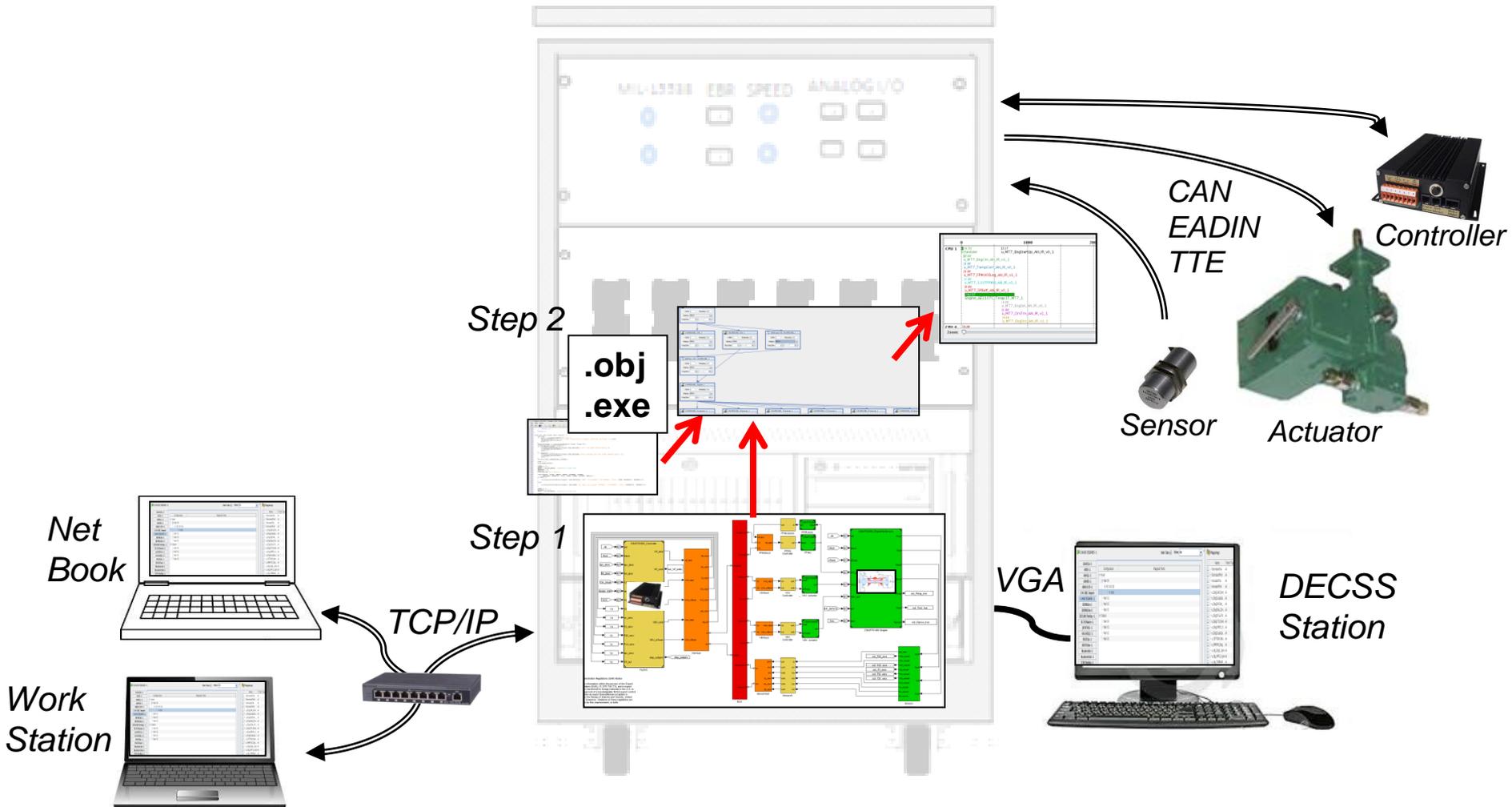
# DECSS Operations



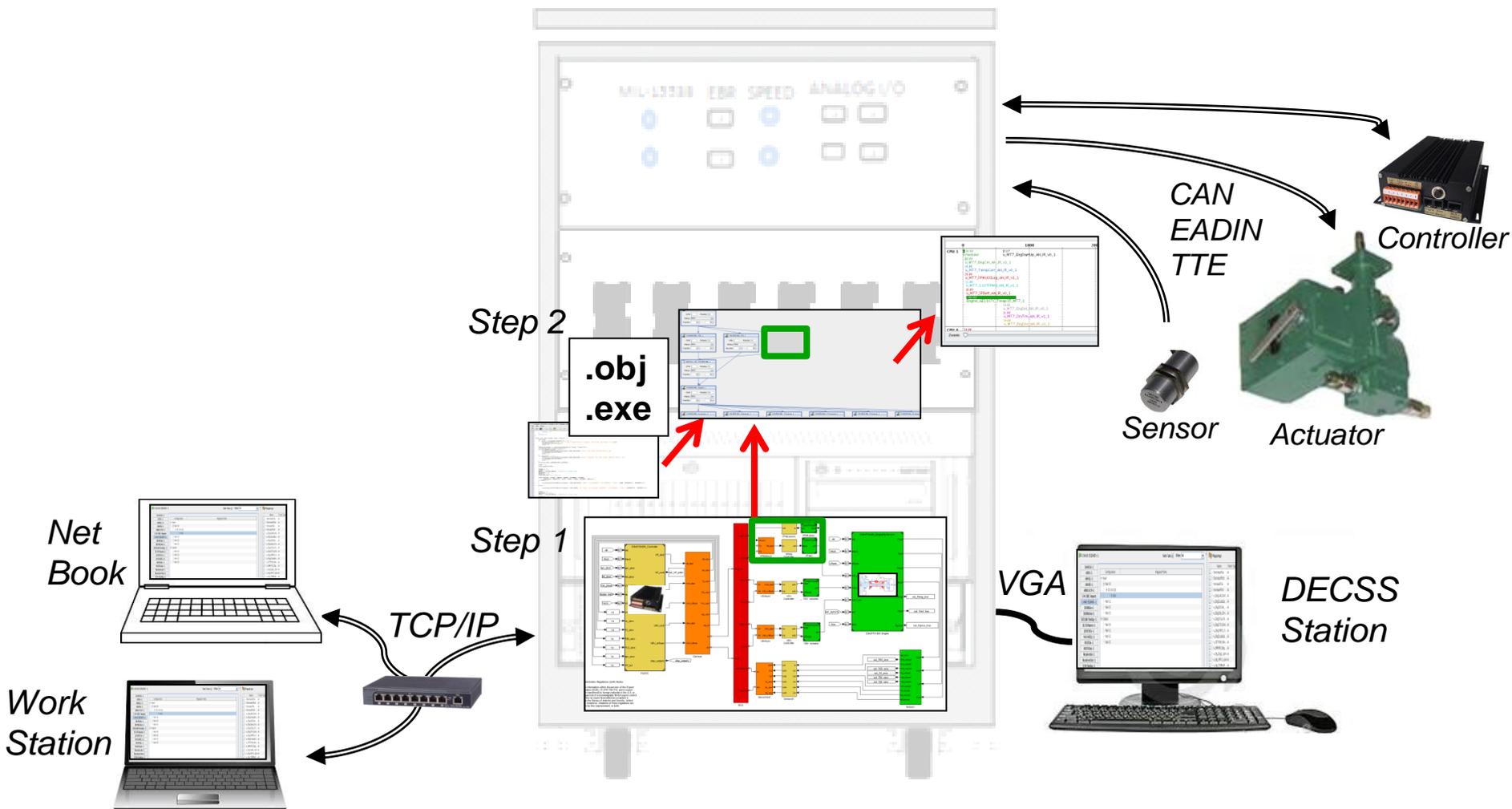
# DECSS Operations



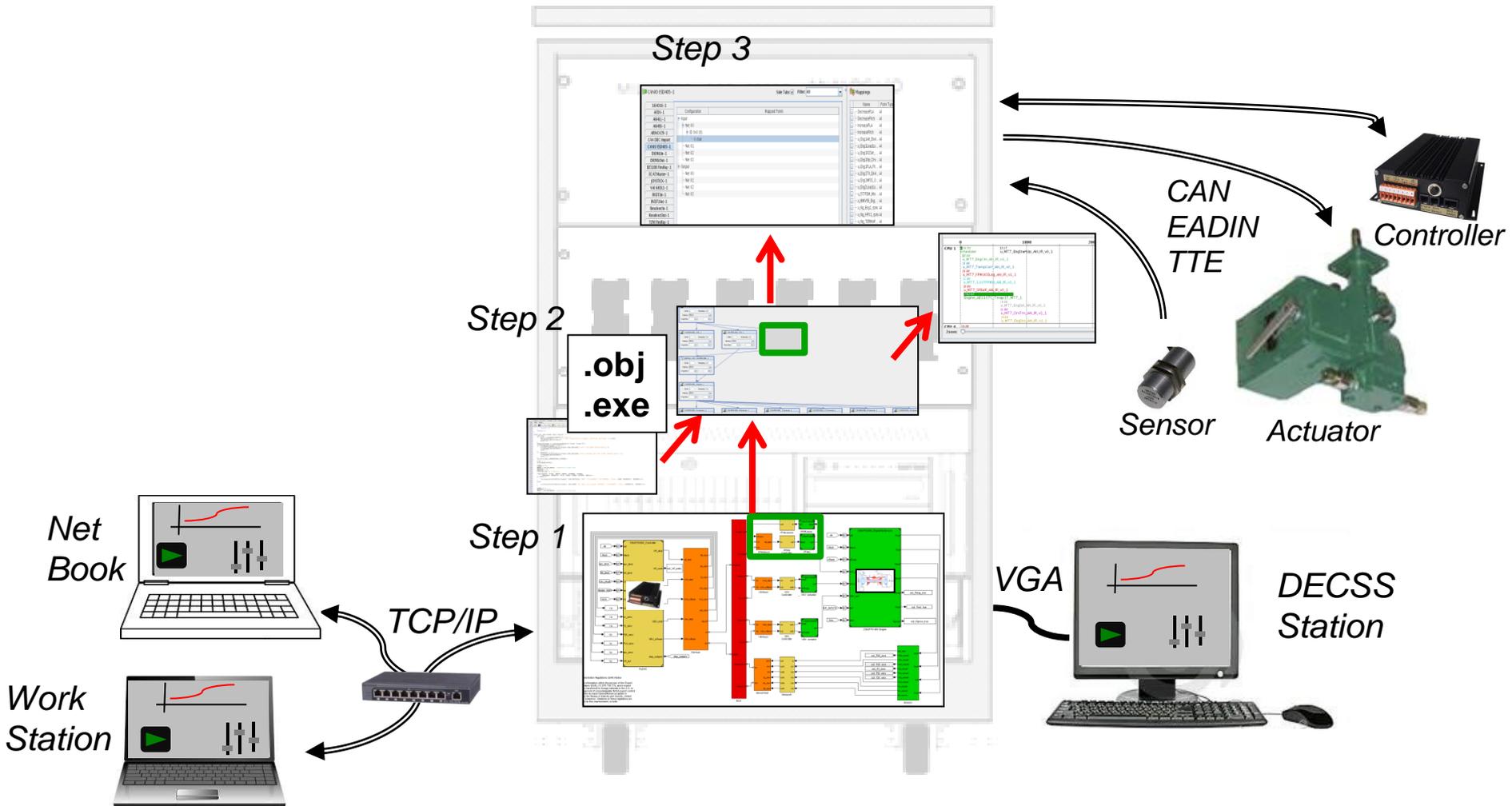
# DECSS Operations



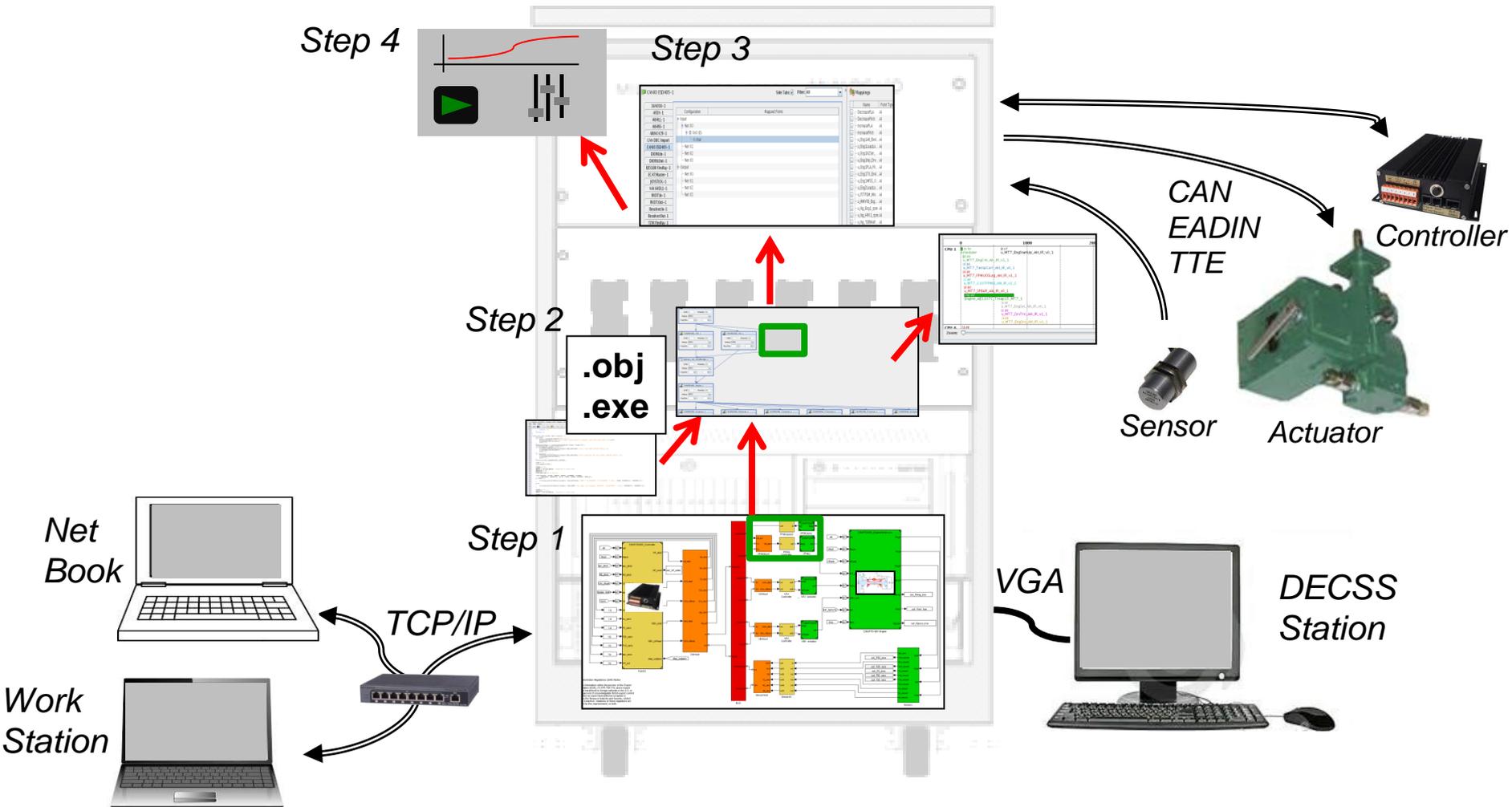
# DECSS Operations



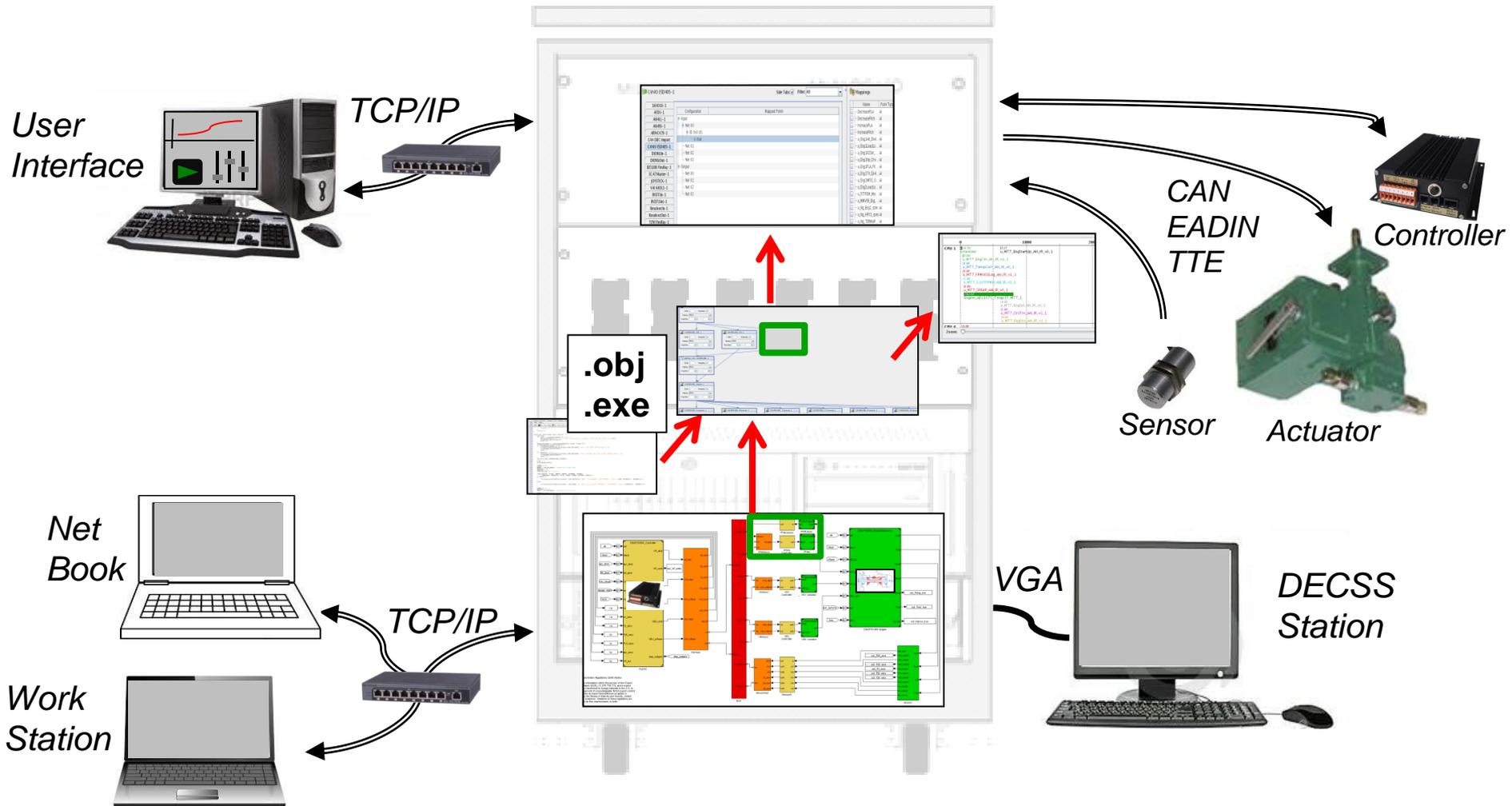
# DECSS Operations



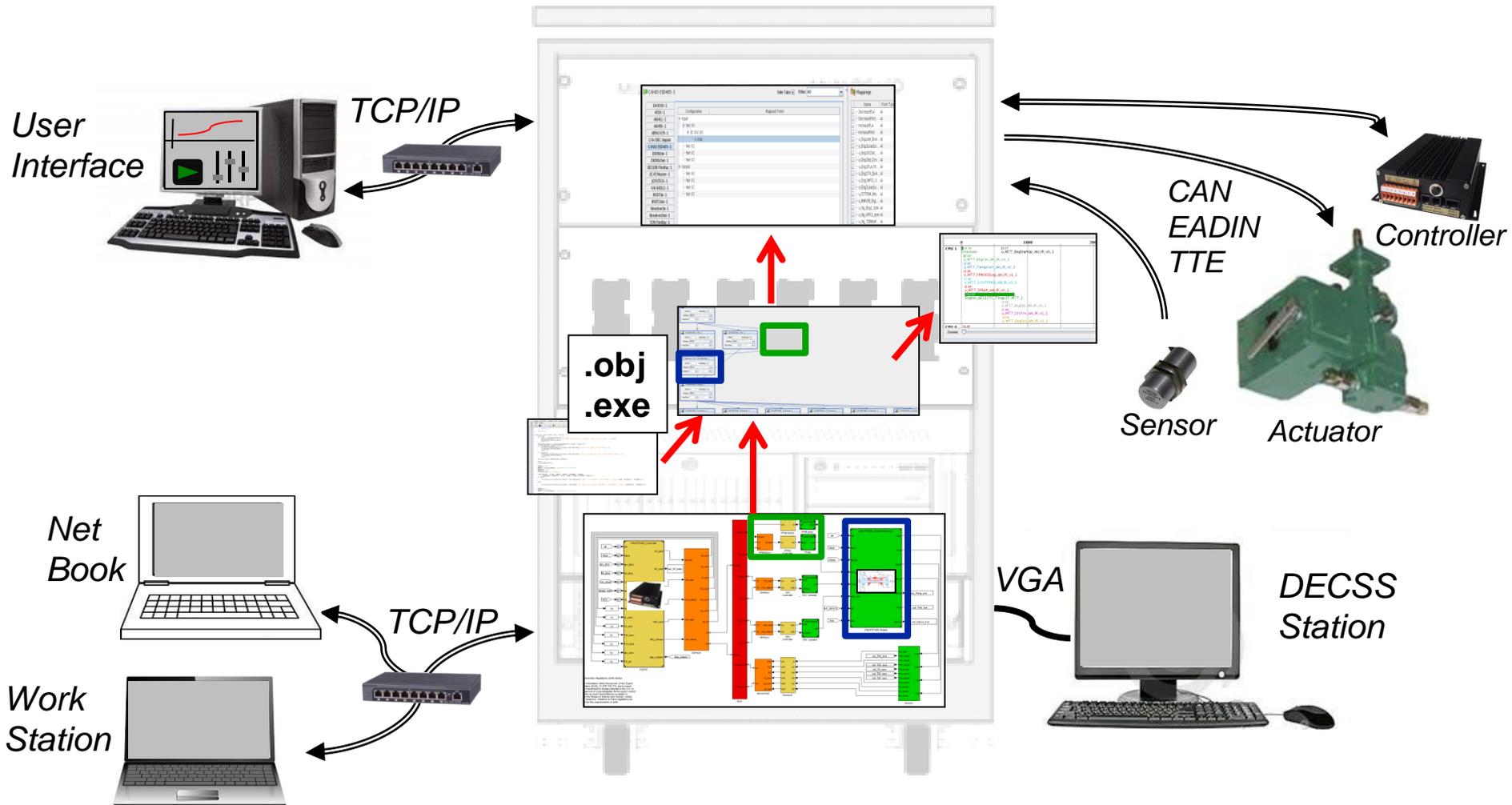
# DECSS Operations



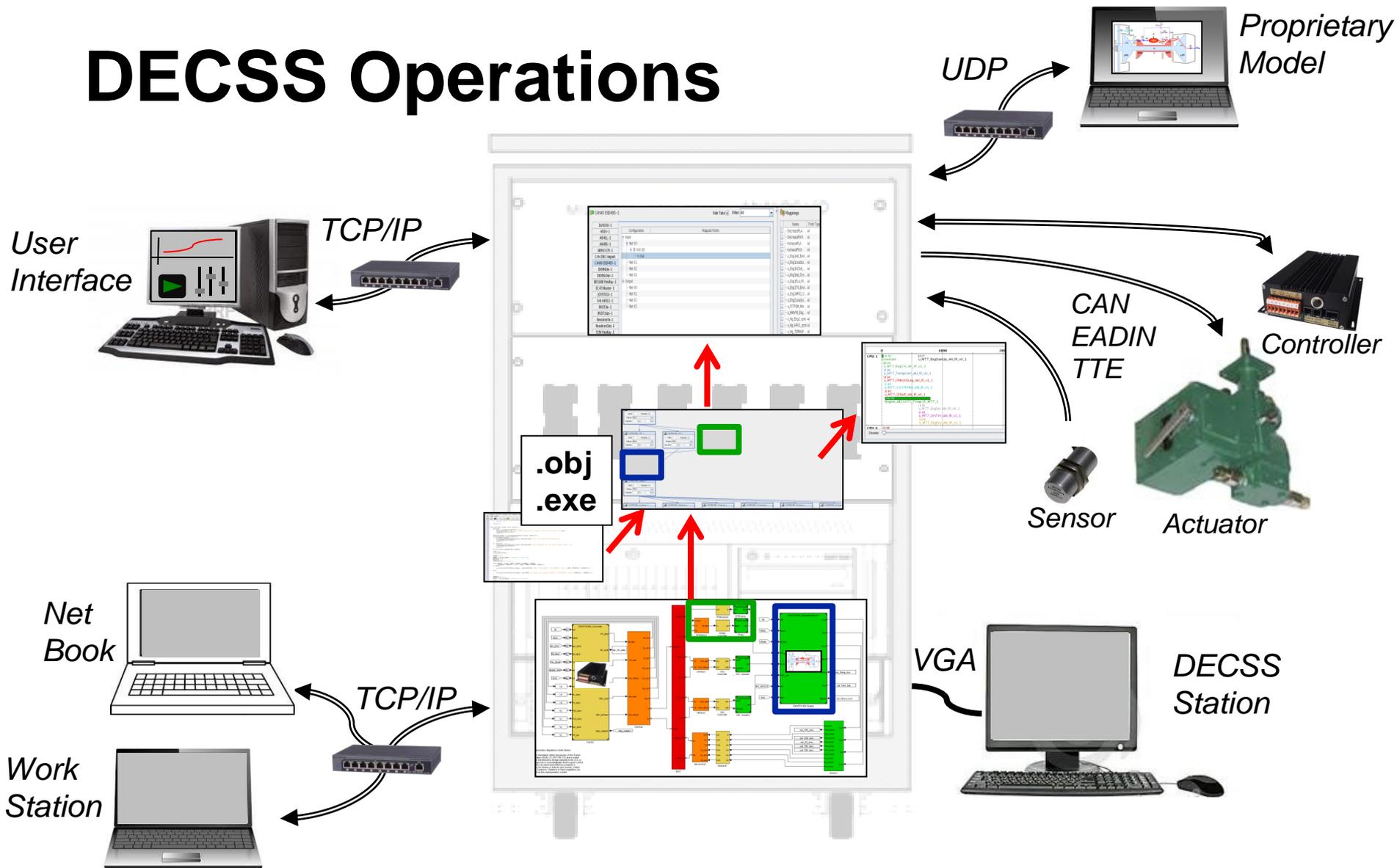
# DECSS Operations



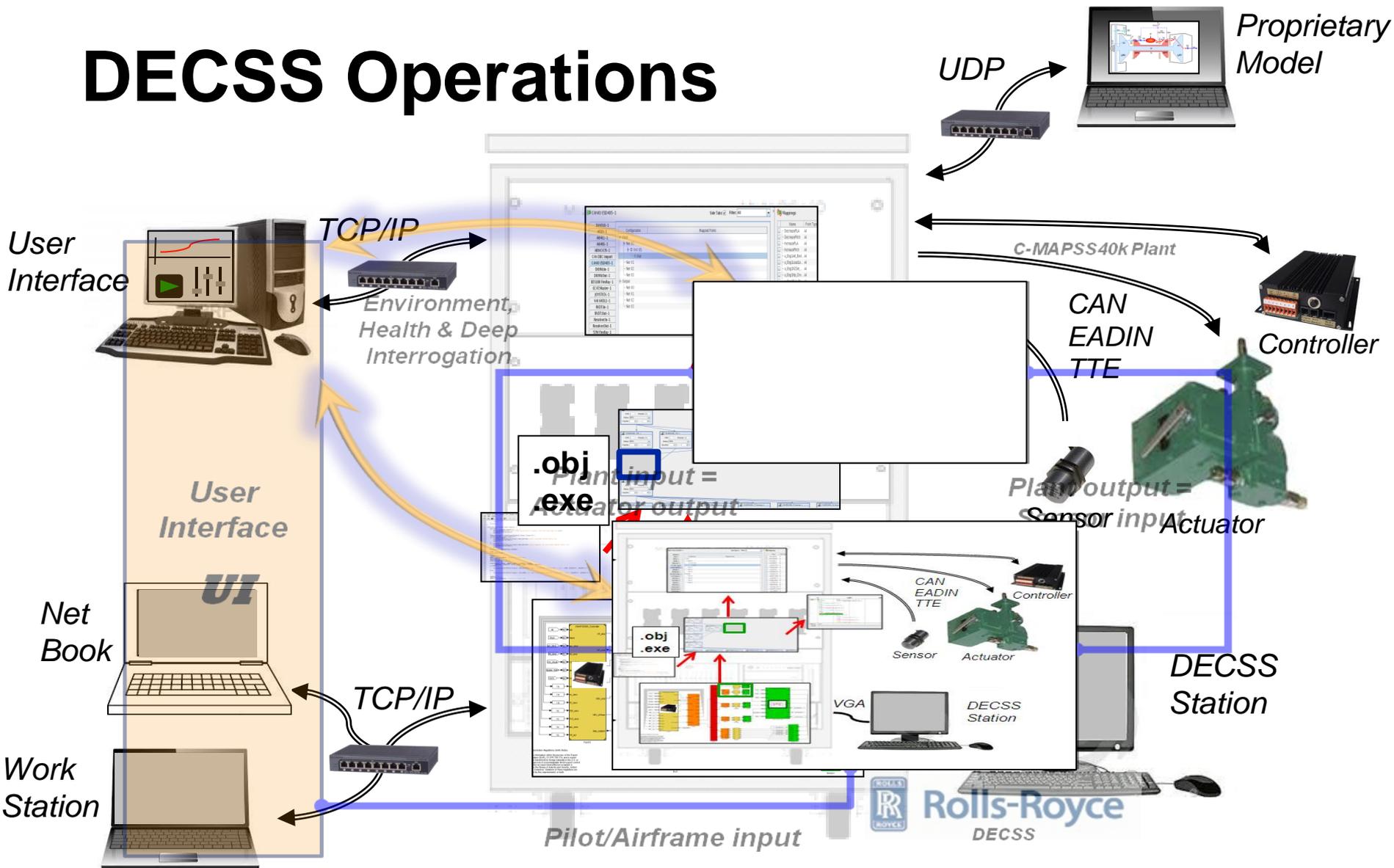
# DECSS Operations



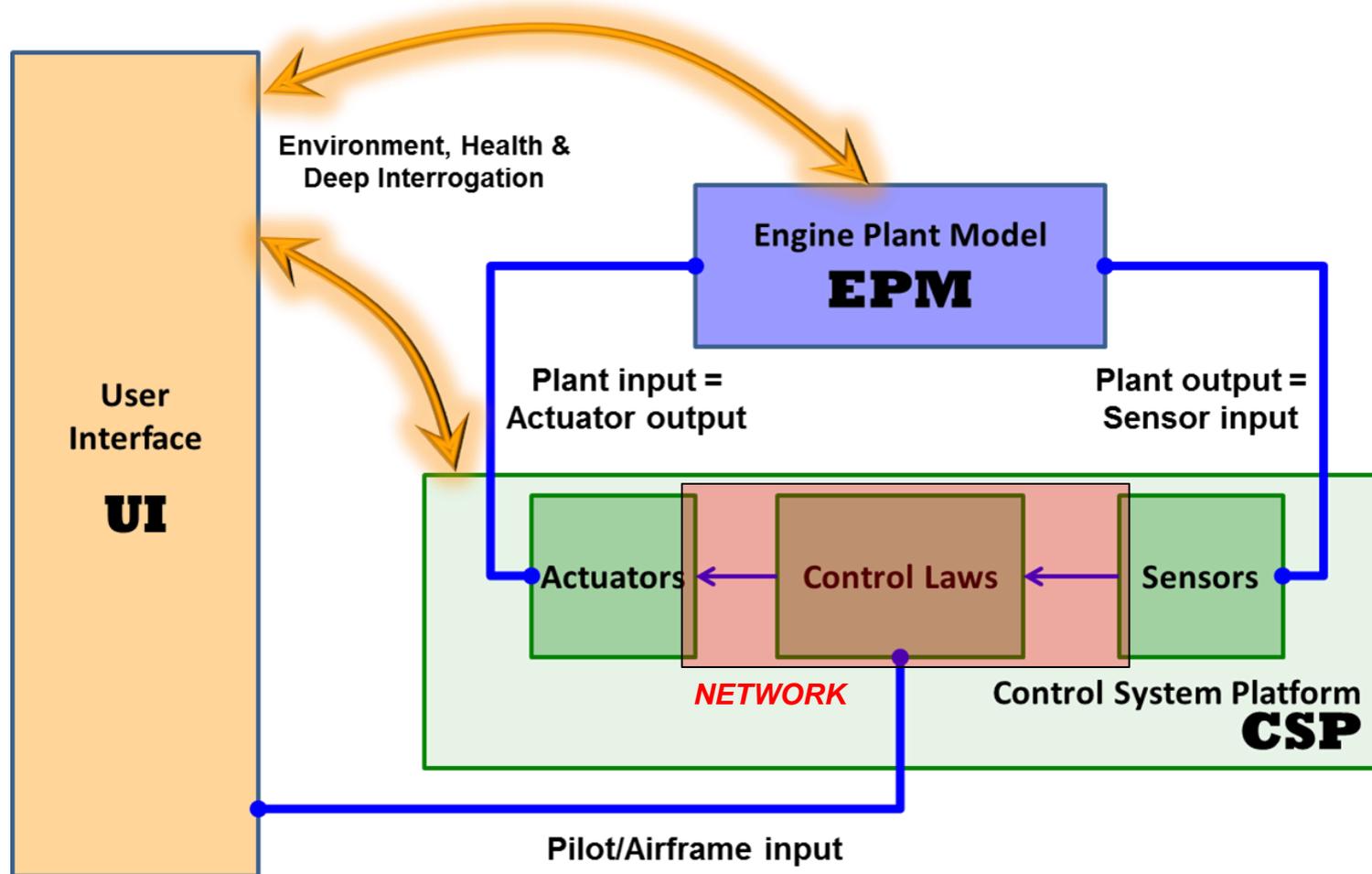
# DECSS Operations



# DECSS Operations



# DECSS Operations



# Summary

- DECSS provides friendly and capable Simulation and HIL test environment
  - | Includes multiple base models to support system development
  - | Monte Carlo testing on simulated system reduces risk
  - | Real-Time HIL
    - Identifies critical limits (of network characteristics)
    - Demonstrates compatibility of sub-systems
  - | Distributed system architecture simplifies interfaces
  - | Flexible network modeling capability with adjustable corruption parameters
  - | Real-time HIL testing of smart components

***DECSS is well suited to support  
Distributed Engine Control System Development***

# Acknowledgements

- NASA Glenn Research Center (GRC)
  - | Controls and Dynamics Branch
- Air Force Research Laboratories (AFRL)
- Distributed Engine Controls Working Group (DECWG)

# Thank you

