Embedded Network Gateway Survivability

Justin Ray, Prof. Phil Koopman

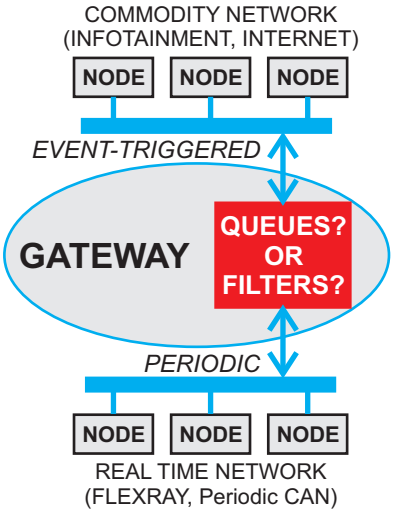
Carnegie Mellon University

How do you mitigate timing faults and timing attacks at the network gateway?

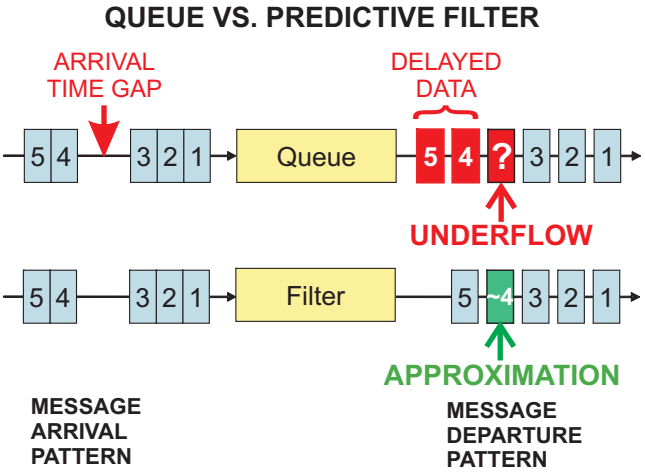
Motivation:

Embedded systems are increasingly connected to the Internet. Usually there is a compelling need to provide an indirection connection between real time control networks (CAN, FlexRay) and infotainment or telematics networks. Clearly a gateway is required to keep problems on the lower criticality networks from spilling onto the real time control networks. But, there is no solid basis for understanding the fundamental question of: ***what goes in the gateway?***

Our research concentrates on the propagation of timing attacks and timing faults across the gateway. A typical gateway design approach is to put in a FIFO Queue to mitigate the effects of message bursts, and drop messages if needed. We are asking questions such as: Do queues work for the gateway? (Queues work poorly.) If queues aren't the answer, what should be used instead? (Perhaps predictive filters.) Does the gateway need to know the semantics of the data it is passing? (Probably it does).



Approach:



Examine difference between Queue and Filter performance:

- Queues leave a gap in outgoing messages, causing a time delay for every message after a gap caused by bunched arrivals
- Filters predict the missing value, then proceed to real values as they arrive at the end of an incoming message clump
- Initial result: queues cause delays that disrupt system dynamics
  Using a queue can be *worse* than doing nothing!

Predictive filters probably need to be customized

- Good filter creates approximation based upon:
  - Type of data (continuous value, enumerated mode, etc.)
  - Time constant of data compared to sampling rate
  - Sensitivity of applications to estimation error
- Gateways will likely need multiple mechanisms and policies AND will need to be configured on a per-message-stream basis

Status:

Current Results:

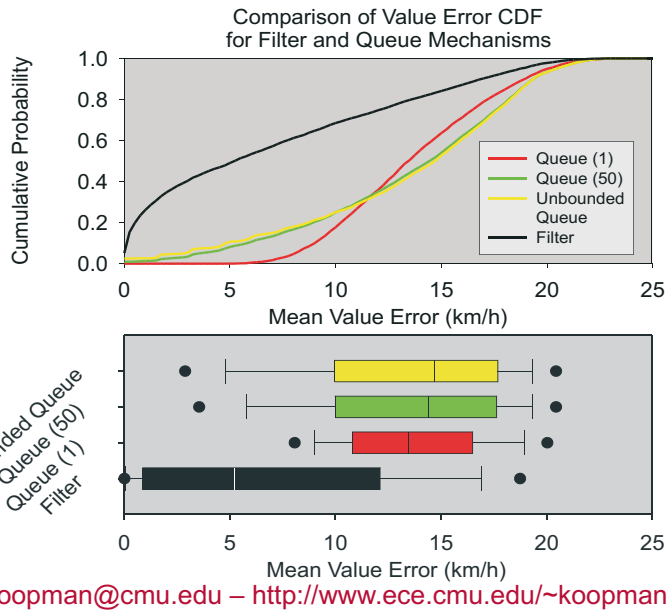
- Simple filters perform better than queues
- Complex filters have problems with fast time constants
- Simple hybrid filters out-perform complex mono-filters

Future Work:

- Good filters for different time constants and data types
- Create hybrid filter + failure detectors
- Formulate generic sets of mechanisms (e.g., filters) and policies (e.g., when does data go stale) to create a set of building blocks for a generic gateway

Timeline:

- Expected Ph.D. proposal summer 2010



Electrical & Computer ENGINEERING

May 2010 – koopman@cmu.edu – http://www.ece.cmu.edu/~koopman