Integrated Timing Analysis & Verification of Component-based Distributed Real-time Systems

Pranav Srinivas Kumar
Advisor: Dr. Gabor Karsai
Institute for Software Integrated Systems
Vanderbilt University

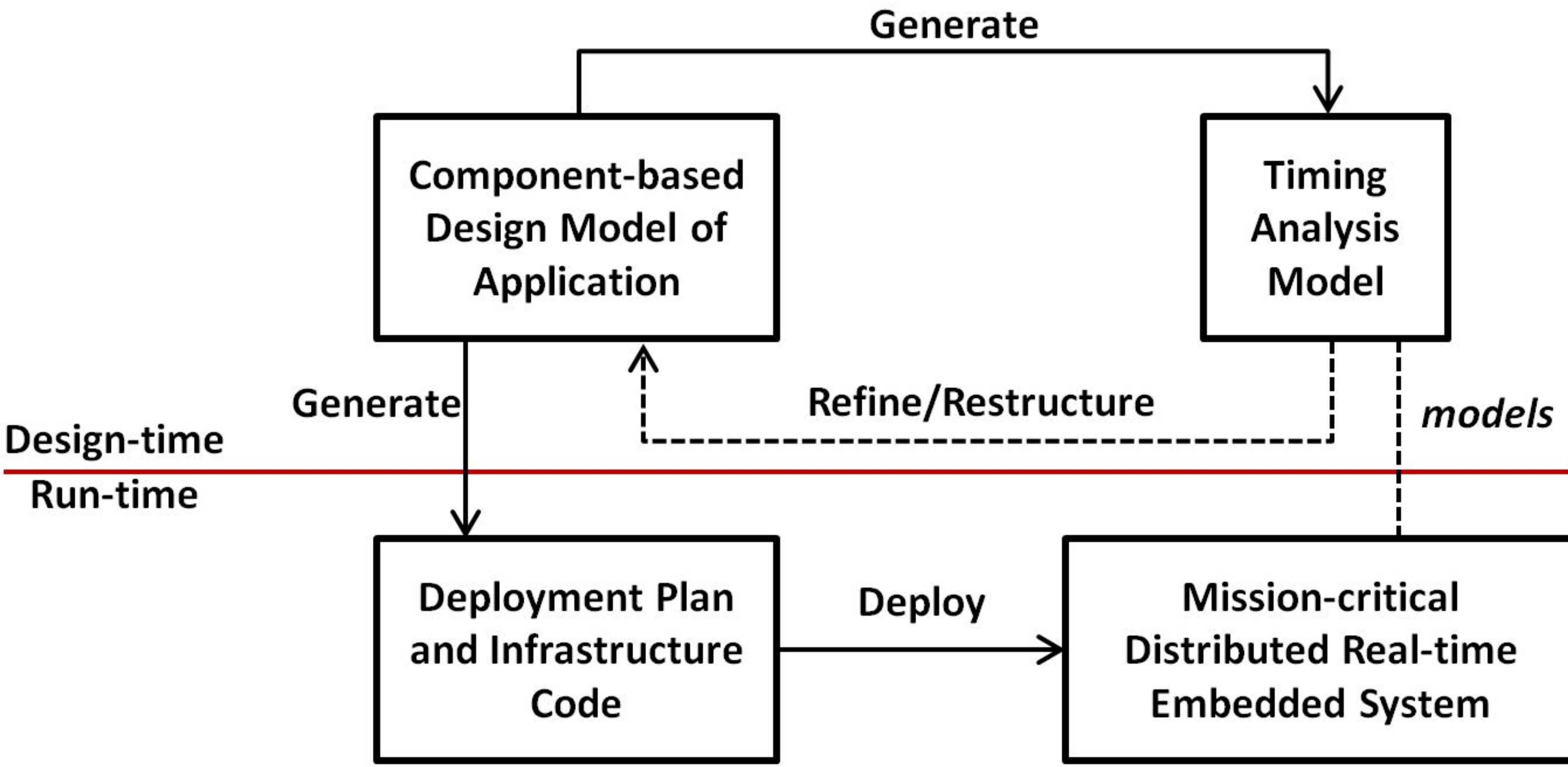
This work was supported by DARPA under contract NNA11AB14C and USAF/AFRL under Cooperative Agreement FA8750-13-2-0050, and by the National Science Foundation (CNS-1035655). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA, USAF/AFRL, or the NSF.

Distributed Real-time Systems

- Avionics, Locomotive control, Industrial Control and Automation
  - Utilizing dependable and predictable software is critical
- Functionality of DRE systems requires large and complex software designs
- Model-driven Development & Component-based Software Engineering
  - Assemble tested components that implement services - Reusability
- Heterogeneous concerns: Strict timing requirements, resilience to faults
- Design-time modeling and analysis methods are a requirement
  - Subscribing to MDD, we design systems by constructing models - We can use these models for analysis

- *https://en.wikipedia.org/wiki/Space_Shuttle_orbiter*
- *https://en.wikipedia.org/wiki/History_of_robots*

Verification-driven Workflow



Analysis Goal: Ensure that a component-based DRE system meets its temporal specifications

Fundamentals

Real-time Systems

- Definition
  - Real-time systems (RT-Systems) are computational systems that offer an assurance of timeliness of service provision
  - Systems whose correctness depends on functional as well as temporal aspects
- Evaluation
  - Timeliness of applications - The execution times of all application tasks must meet individual deadlines
- Distributed real-time systems are characterized by a set of computational nodes that interact by exchanging messages

Timing Analysis of Real-Time Systems

- Assume an ideally functional real-time system
- Annotate computational tasks with timing properties
  - Worst-case Execution Time (WCET)
- Characterize the system using an Analysis Model to study the timing behavior of the entire system
- *Verification*: Prove that the implementation of the system is correct w.r.t. requirements
  - Does our product comply with regulations?
- *Validation*: Show that the implementation of the system meets user's expectations
  - Does our product meet the needs of the customer?

Analysis Challenges

- With CBSE, tried and tested components are composed to construct large and complex systems
- *Principle of Compositionality*
  - Properties of a composed system can be derived from the properties of its components and connections
- *Challenge*
  - Ensure the correct and timely operation of a system consisting of a composed set of interacting components
- *Timing Analysis Requirements*
  - Verify the timing properties of each component
  - Analyze the temporal behavior of a component assembly

Related Research

General Methodologies

- *Testing and Evaluation*
  - Black box and White box Testing
  - Manual and Automatic test input generation
  - Simulation-based Testing
- *Static Code Analysis*
  - Sophisticated methods to analyze executable code
  - Language restrictions and skillset requirements
- *Formal Analysis Methods* - Check correctness with mathematical rigor
  - *Theorem Proving*
    - Requires human intervention and expertise
  - *Model Checking*
    - Requires precise specification of properties to be examined
    - Less suitable for data intensive applications

State Space Analysis

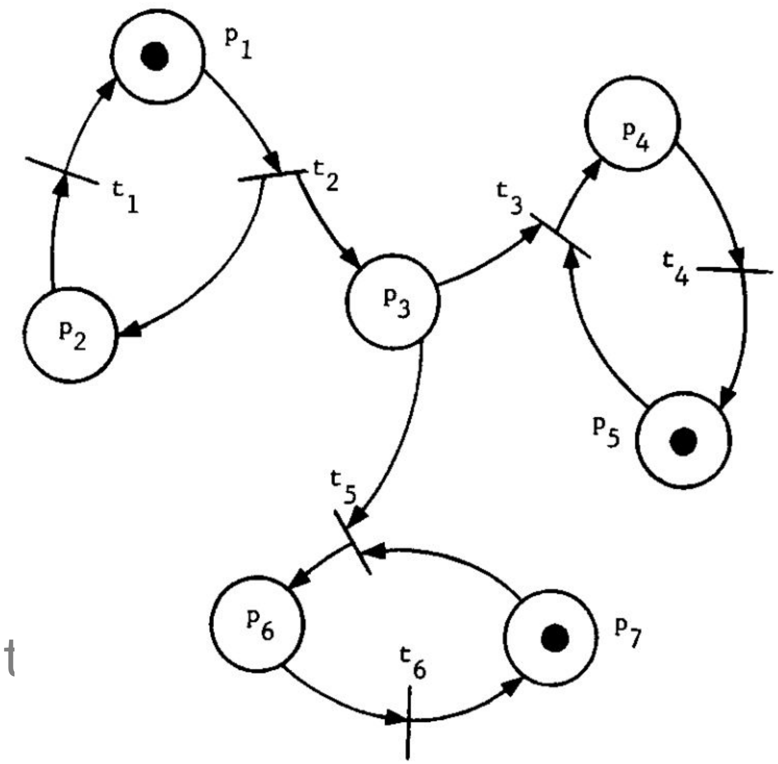
- Suitable for automatic analysis and verification of the behavior of the system
- We construct a structure that consists of all states that a system can reach, and all transitions that system can make between these states - *state space*
- State Space Explosion
  - Size of the state space tends to grow exponentially in the number of its processes and variables
  - Methods have been developed to reduce the number of required states for answering analysis questions
- The user may ask various queries on the state space and obtain various statistics on the state space

Some Timing Analysis Tools and Frameworks

- Cheddar Real-Time Scheduling Framework *[Singhoff et al., 2004]*
  - Ada framework based on real-time scheduling theory
  - Scheduling simulation and feasibility tests
    - Compute schedules and look for task constraint properties in the computed schedule
    - Classic real-time schedulers - RMS, EDF etc.
- UPPAAL *[Bengtsson et al., 1996]*
  - RT systems modeled as a network of Timed Automata
    - Finite state machines extended with clock variables
    - Clocks are real-valued and progress synchronously
  - Can formulate schedulability analysis problems that UPPAAL can check with its model checker

Petri Net-based Analysis

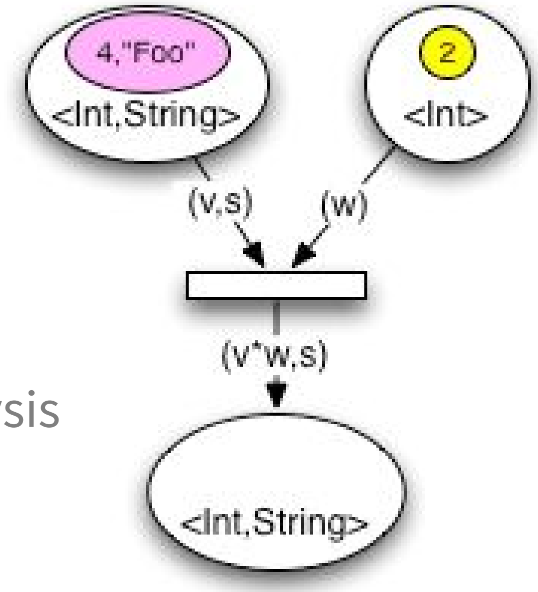
- Formal model of information flow
- Places
  - Places contain tokens
  - Input Places and Output Places
- Transitions
  - Transitions represent events
  - Firing rules - input places with tokens
- Petri net Execution - Token Movement
- Modeling Dynamic System Behaviors
  - Concurrency, Synchronization and Resource Sharing



J. L. Peterson. Petri nets. ACM Computing Surveys (CSUR), 9(3):223–252, 1977.

High-level Petri Nets

- Timed Petri Nets
  - Occurrence times associated with transitions
  - Tokens are removed from input places at the beginning of the occurrence period and deposited to output places at the end of this period
- Colored Petri Nets
  - Tokens can be complex typed data structures
  - Richer constraints on transitions
  - Compact hierarchical description
  - CPN Tools for simulation and state space analysis *[Ratzer et al., 2003]*



http://scienceblogs.com/goodmath/2007/10/10/colored-petri-nets/

Analyzing AADL Models with Petri nets

- *[Renault et al., ISORC 2009]* Translate AADL models to Petri nets
    - subnet generated for each AADL *thread* - Nets become intractable for large applications - does not scale well for large process sets
    - Tightly bound to AADL modeling concepts
- *[Renault et al., RSP 2009]* Timed Petri net to model thread lifecycle
    - State of AADL threads - modeled using places
    - Lifecycle over the states - handled by transitions
    - Observer places to monitor Petri net execution
        - Detect property violations and operation completion
    - Only considers periodic threads with fixed execution time
    - No preemptive scheduling

Modeling and Analysis Suite

- *[Gonzalez et al., 2001]* Timing analysis integration for fixed priority scheduled systems - Single processor and distributed systems
- Event-driven model with complex dependence patterns
- No model checking or formal verification
- MAST events are periodic, sporadic or bursty
  - Does not model the nature of the event itself e.g. blocking behavior
- Provision to model schedulable entities and processing resources
  - No explicit provision to model hierarchical schedulers - primitive support now in MAST 2

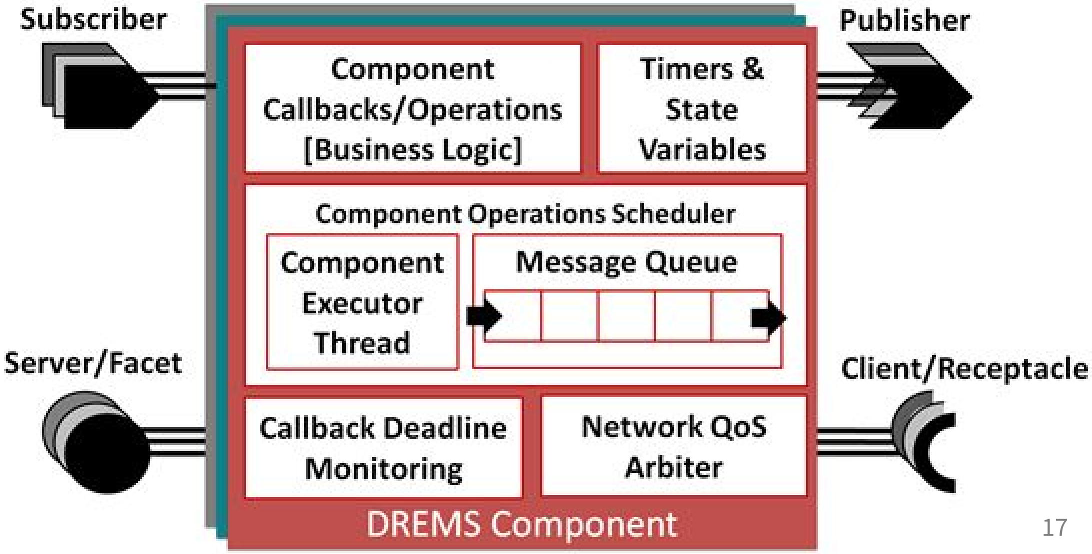
Design Model: DREMS

[Levendovszky et al., 2014]

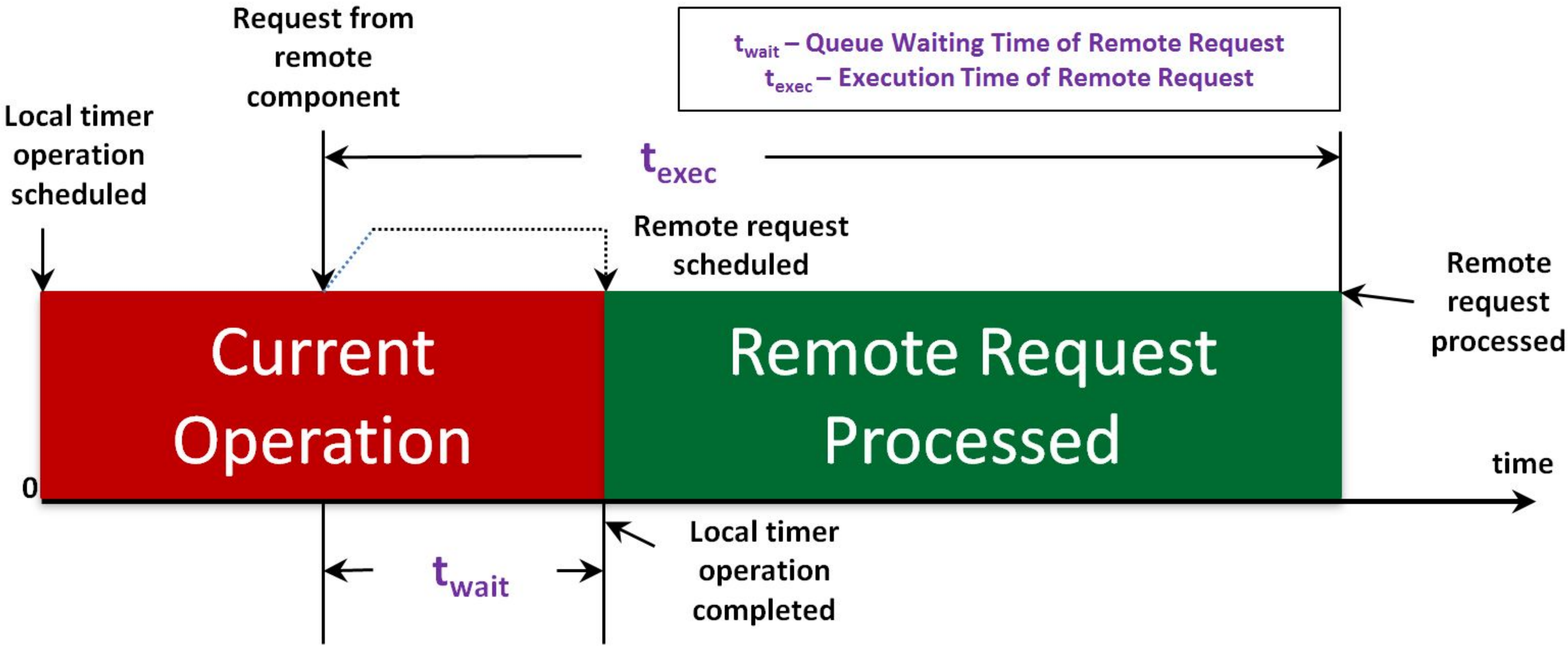
Distributed Real-Time Managed Systems

- Design, Implementation, and Operation of DRE systems
- Model-driven development
- Component-based Software Engineering Principles
    - Varied and generic interaction patterns
    - Precise component execution semantics
- Robust runtime platform
- Managing mixed-criticality real-time applications
- Enforces strict timing requirements

http://www.wired.com/images_blogs/dangerroom/2013/05/System-F6-Satellite_formation_landscape.jpg



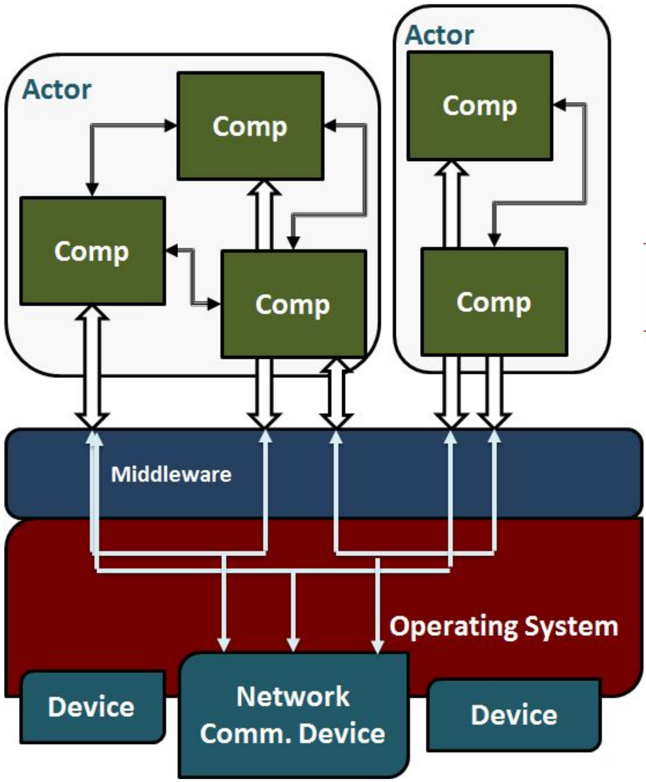
Component Model - Execution Semantics

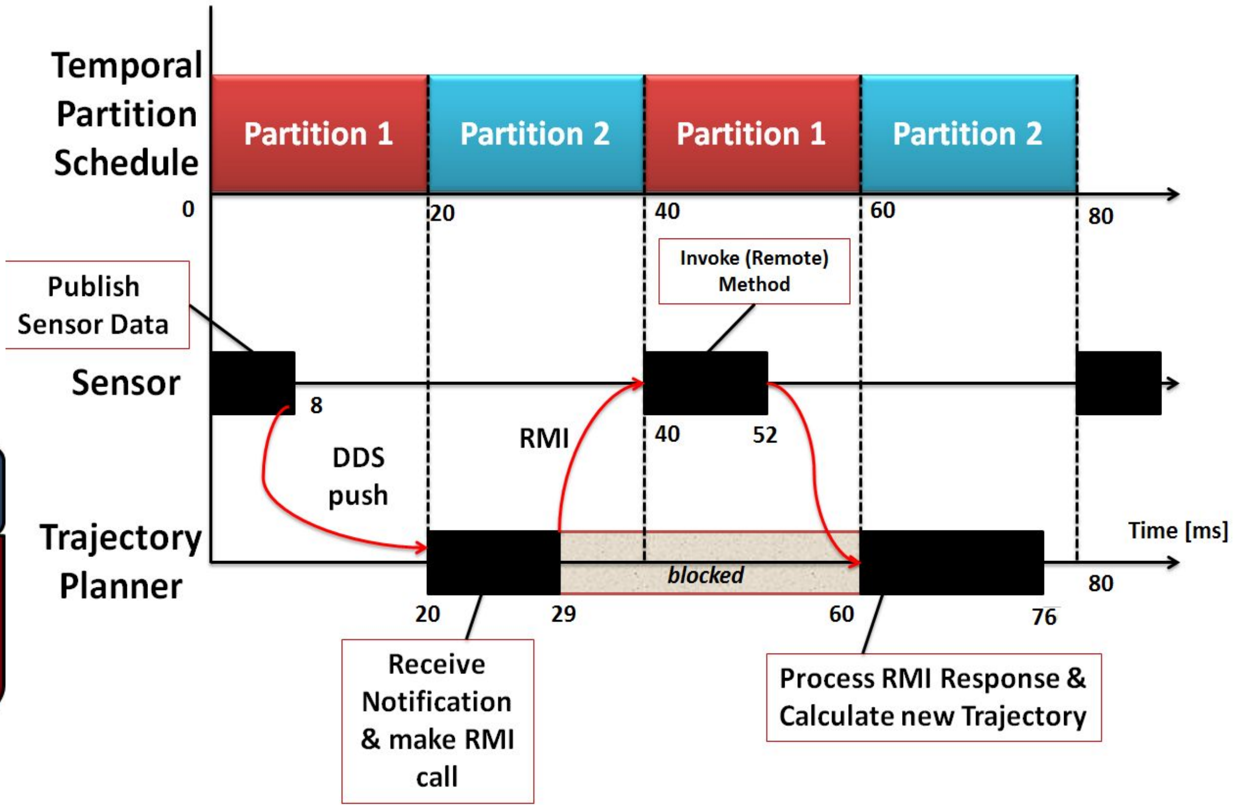


- A single component executor thread executes operation requests scheduled in the message queue
- Operation scheduling is non-preemptive i.e. the next request is processed only when the current operation is completed

DREMS OS - Temporal Partition Scheduler

- ARINC-653 style temporal and spatial partitioning
- DREMS components are grouped into uniquely identifiable processes - *actors*
- Each actor is assigned to a temporal partition
    - Periodic fixed intervals of the CPU's time
- Temporal isolation is achieved between threads in different partitions





Problem Statement

- We have DREMS-style component-based applications distributed and deployed on a cluster of embedded devices
- Analysis Assumptions
  - Design Model: Component Definitions, assembly, temporal partitioning, deployment model etc.
  - Each component operation executes a sequence of computational steps (of finite duration)
  - Known worst-case estimated time taken by step
- Temporal Behavior of the composed system must meet end-to-end timing requirements
  - Deadlines, Trigger-to-Response times etc.

Research Plan:
Design-time Timing Analysis of Component-based DRE Systems

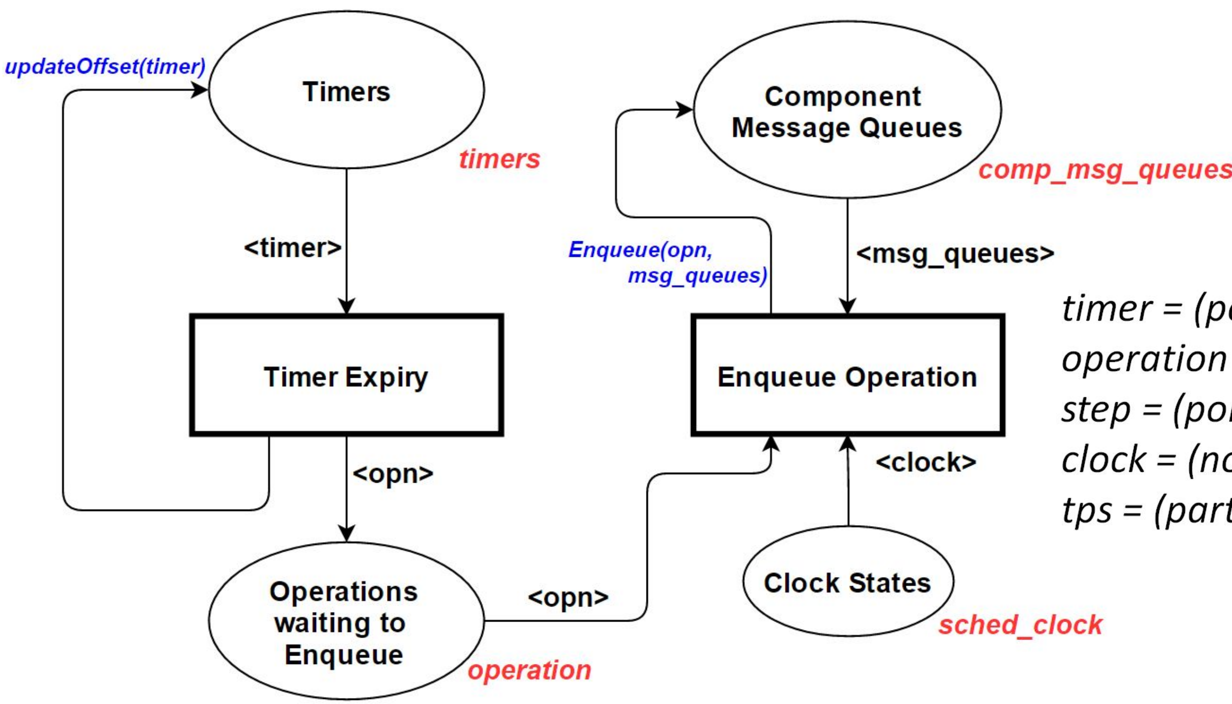
Preliminary Results and Proposed Work

Colored Petri Net-based Timing Analysis

- Challenge
  - How to design and construct an extensible, scalable timing analysis model for component-based DRE systems?
- Approach
  - Design and implement a CPN-based timing analysis model
    - Capture structure and behavior of component assembly
    - Capture business logic of component operations (steps)
    - Capture timing properties of steps in each operations
- Preliminary Results
  - Developed an extensible CPN analysis model
  - Performed bounded *state space analysis* on a variety of DREMS scenarios
    - Detect timing violations e.g. deadline miss, deadlocks etc.
    - Estimate worst-case trigger to response times
    - Estimate processor utilization
    - Estimate partial thread execution orderings
  - Implemented a DREMS *model interpreter* that generates our CPN analysis model from the design model

Colored Petri Net-based Timing Analysis

- Places manage application state
- Transitions represent events
- When transitions fire:
  - Tokens in input places are consumed/modified
  - Output places receive tokens - postconditions of the triggered event
- Output arc inscriptions are able to operate on the consumed tokens



timer = (period, offset, operation)
operation = (opn_name, priority, deadline, steps)
step = (port_name, unblk_list, duration)
clock = (node_name, clock_value, tps)
tps = (partition_name, period, duration, offset)

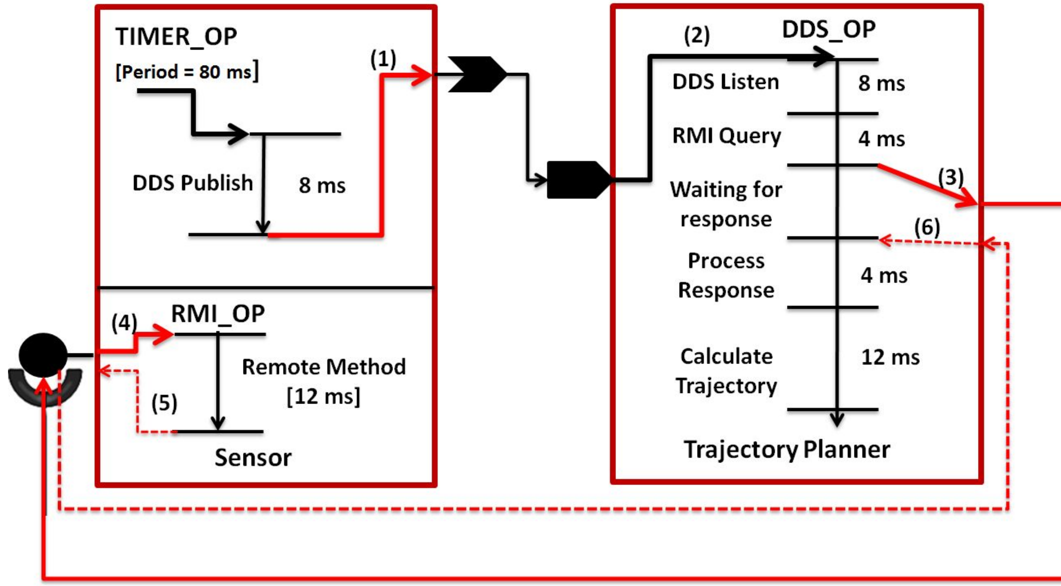
Modeling Component Operations

- Challenge
  - Operation Business logic: Piece of code executed when a component operation is scheduled
  - This code directly affects the behavior of components
  - It is not sufficient to annotate models of component operations with a single WCET - Need a finer grain model of temporal behavior
- Approach
  - Component Operations are modeled and represented as a sequence of timed steps - each step is annotated with a WCET
- Preliminary Results
  - Designed a modeling grammar to represent the business logic of component operations
  - Relevant ports and timers in every component are annotated with operation business logic models
  - When the design model is parsed, these BL models are translated into *CPN tokens* and generated into distinct places in our analysis model

Modeling Component Operations

```
// Trajectory Planner - Subscriber Operation
do DDS_OP {
   LOCAL 12;
   RMI Fetch_Sensor_Data.server_operation;
   LOCAL 16;
}
```

ANTLR-based business logic model is translated into CPN tokens



(* Generated CPN Token for Trajectory Planner Subscriber Operation *)

```
1`[{opn={opname="DDS_OP", op_node="Sat1", op_comp="Trajectory_Planner",
        op_pn="Part2", op_prio=60, op_dl=50, op_st=0, op_dv=false,
        op_steps=[{port_type="Local", port_name="Local", unblk_list=[], induced=false, call_st=0,
                        call_et=0, call_exec_t=0, call_dur=12},
                  {port_type="RMI_Receptacle", port_name="Fetch_Sensor_Data", unblk_list=[],
                        induced=false, call_st=0, call_et=0, call_exec_t=0, call_dur=0},
                  {port_type="Local", port_name="Local", unblk_list=[], induced=false,
                        call_st=0, call_et=0, call_exec_t=0, call_dur=16}]}}]
```

Modeling and Analysis Improvements

- Challenge - Alleviate state space explosion - Caused by:
  - Our model of time
  - Our model of Distributed Deployment
- Approach
  - Modify our timing analysis execution by introducing dynamic time progression (time jumps)
  - Remodel distributed deployments - use ordered lists instead of unordered sets
- Preliminary Results
  - CPN execution progresses at a much faster rate
    - Reduces number of states recorded in the state space
  - Reduces the number of transition firings with regards to distributed deployments
      - from C! to 1 for C computing nodes
      - Can support a larger number of computing nodes with improved scalability

Investigating Advanced State Space Analysis

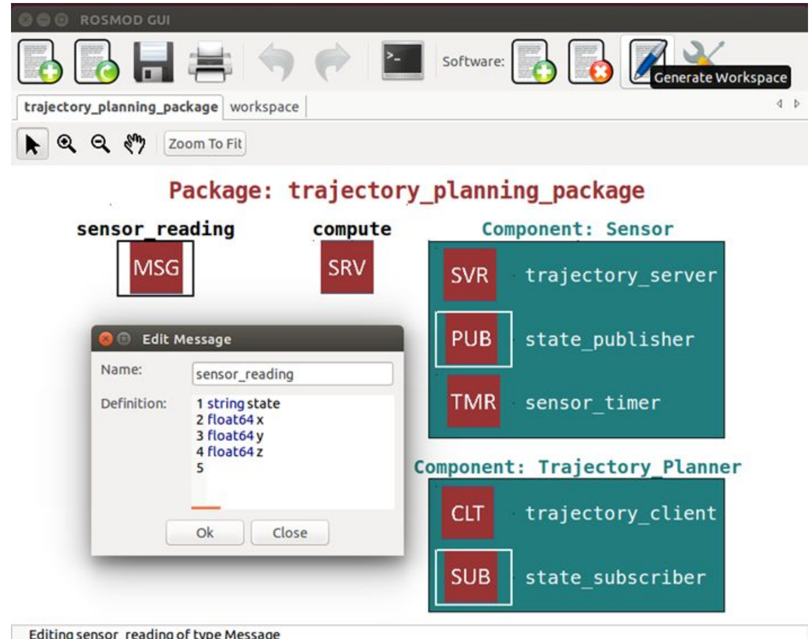
- Challenge
  - How to improve the efficiency and speed of state space generation and leverage advanced analysis and memory management techniques
- Approach
  - ASAP CPN Analysis Tool - Integrate our CPN analysis model with ASAP platform and apply advanced analysis methods
- Preliminary Results
  - Combat State Space Explosion - Sweep-line method
  - Safety Properties, Deadlock, Liveness, Boundedness, …
  - On-the-fly efficient verification of system constraints
  - Time taken to generate state space is greatly reduced by applying ASAP memory optimization, hashing methods, and on-the-fly checking

State Space Exploration Time (seconds)

Model	States	CPN Tools	ASAP	Speed-up
50 component DREMS sample	124K	846	211	4.01
100 component DREMS sample	485K	2,160	576	3.75

Experimental Validation

- Challenge
  - How to validate theoretical results obtained from our CPN analysis methodology?
- Approach
  - RCPS Testbed for Validation
  - Execute components individually with RT priority
  - Obtain worst-case *pure execution time* of operations
    - Low-overhead logging to hook into middleware
  - Construct business logic models and generate CPN
  - Analyze and Verify
  - Compare CPN results with real-system execution



Analysis of Long-running Operations (LRO)

- Challenge
  - How to integrate and support AI-style long-running operations with DREMS?
  - Our DREMS component model implements a non-preemptive operations scheduler - LRO can disrupt the execution of highly critical operations
  - How to model the execution semantics of LRO?
- Approach
  - We assume known fixed-interval LRO cancellation *checkpoints*
  - We also assume that the LRO e.g. path-planning search algorithm reaches a safe state at each checkpoint before cancelling LRO
  - This way, a higher-priority operation waiting in the component message queue would cause LRO cancellation before its own execution
  - This requires the introduction of new concepts to our operation steps e.g. checkpointing steps

Analysis of Cyber-Physical Systems (CPS)

- Challenge
  - How to support the analysis of component-based applications interacting with a physical environment e.g. UAV swarms?
    - Physics Simulation
  - How to model component interactions with simulation interface?
  - How to validate our theoretical analysis results?
- Approach
  - We will model and analyze a variety of interaction patterns between ‘I/O components’ and CPS sensors/actuators
    - Sensors are modeled as periodic data pumps
    - Actuator interfaces are modeled as non-blocking fixed-time operations
  - Analysis results will be evaluated against prototype CPS deployments in our RCPS testbed

Summary and Schedule

- We have proposed an extensible, scalable CPN-based timing analysis methodology for component-based DRE systems
  - Precise model of component execution semantics
  - Tool support for advanced analysis and verification
  - Interesting potential extensions and scope
- We have presented some remaining work in this topic and outlined our plans for completion

TASKS	PROPOSED TIMELINE
Thorough Experimental Validation	**10/2015 - 11/2015**
Analysis of Long-Running Operations	**10/2015 - 12/2015**
Modeling and Analysis of CPS deployments	**11/2015 - 02/2016**
Dissertation Writing	**10/2015 - 03/2016**

Publications

- **P. S. Kumar**, A. Dubey, and G. Karsai. Colored petri net-based modeling and formal analysis of component-based applications. In 11th Workshop on Model Driven Engineering, Verification and Validation MoDeVVa 2014, page 79, 2014
- **P. S. Kumar** and G. Karsai. Integrated analysis of temporal behavior of component-based distributed real-time embedded systems. In Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2015 IEEE International Symposium on Real-time Computing (ISORC), pages 50–57, April 2015
- **P. Kumar**, W. Emfinger, A. Kulkarni, G. Karsai, D. Watkins, B. Gasser, C. Ridgewell, and A. Anilkumar. ROSMOD: A Toolsuite for Modeling, Generating, Deploying, and Managing Distributed Real-time Component-based Software using ROS. In Proceedings of the IEEE Rapid System Prototyping, RSP 2015, Amsterdam, Netherlands, 2015. IEEE
- **P. Kumar**, W. Emfinger, and G. Karsai. A Testbed to Simulate and Analyze Resilient Cyber-Physical Systems. In Proceedings of the IEEE Rapid System Prototyping, RSP 2015, Amsterdam, Netherlands, 2015. IEEE
- W. Emfinger, **P. Kumar**, A. Dubey, W. Otte, A. Gokhale, G. Karsai. DREMS: A Toolchain for the Rapid Application Development, Integration, and Deployment of Managed Distributed Real-time Embedded Systems. In Proceedings of the IEEE Real-Time Systems Symposium, RTSS@Work 2013, Vancouver, Canada, 2013. IEEE
- Balasubramanian, D., W. Emfinger, **P. S. Kumar**, W. Otte, A. Dubey, and G. Karsai. An application development and deployment platform for satellite clusters. In Proceedings of the Workshop on Spacecraft Flight Software, 2013
- Balasubramanian, D., A. Dubey, W. R. Otte, W. Emfinger, **P. Kumar**, and G. Karsai. A Rapid Testing Framework for a Mobile Cloud Infrastructure. In Proceedings of the IEEE International Symposium on Rapid System Prototyping, RSP, 2014. IEEE
- D. Balasubramanian, A. Dubey, W. Otte, T. Levendovszky, A. Gokhale, **P. Kumar**, W. Emfinger, and G. Karsai. Drems ml: A wide spectrum architecture design language for distributed computing platforms. Science of Computer Programming, 2015
- Levendovszky, T., A. Dubey, W. R. Otte, D. Balasubramanian, A. Coglio, S. Nyako, W. Emfinger, **P. Kumar,** A. Gokhale, and G. Karsai. DREMS: A Model-Driven Distributed Secure Information Architecture Platform for Managed Embedded Systems. In IEEE Software, vol. 99: IEEE Computer Society, 2014. IEEE
- AWAITING REVIEW:
  - W. Emfinger, **P. Kumar**, A. Dubey, G. Karsai. Towards Assurances in Self-Adaptive, Dynamic, Distributed Real-time Embedded Systems. In Software Engineering for Self-Adaptive Systems: Assurances, 2015.

References

- ***[Singhoff et al., 2004]*** A flexible real time scheduling framework. In Proceedings of the 2004 Annual ACM SIGAda International Conference on Ada: The Engineering of Correct and Reliable Software for Real-time &Amp; Distributed Systems Using Ada and Related Technologies, SIGAda ’04, pages 1–8, New York, NY, USA, 2004. ACM.
- ***[Bengtsson et al., 1996]*** UPPAAL - a tool suite for automatic verification of real-time systems. Springer, 1996.
- ***[Ratzer et al., 2003]*** Cpn tools for editing, simulating, and analysing coloured petri nets. In Proceedings of the 24th International Conference on Applications and Theory of Petri Nets, ICATPN’03, pages 450–462, Berlin, Heidelberg, 2003. Springer-Verlag.
- ***[Renault et al., RSP 2009]*** Adapting models to model checkers, a case study : Analysing aadl using time or colored petri nets. In Rapid System Prototyping, 2009. RSP ’09. IEEE/IFIP International Symposium on, pages 26–33, June 2009.
- ***[Renault et al., ISORC 2009]*** From aadl architectural models to petri nets: Checking model viability. In Object/Component/Service-Oriented Real-Time Distributed Computing, 2009. ISORC ’09. IEEE International Symposium on, pages 313–320, March 2009.
- ***[Gonzalez et al., 2001]*** Mast: Modeling and analysis suite for real time applications. In Real-Time Systems, 13th Euromicro Conference on, 2001., pages 125–134, 2001.
- ***[Levendovszky et al., 2014]*** Distributed real-time managed systems: A model-driven distributed secure information architecture platform for managed embedded systems. IEEE Software, 31(2):62–69, 2014.