NETWORK PERFORMANCE ANALYSIS AND NETWORK MANAGEMENT FOR CYBER-PHYSICAL SYSTEMS AND THEIR APPLICATIONS

By

William A. Emfinger

Proposal

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

May, 2015

Nashville, Tennessee

Approved: Date:

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

And here is my dedication.

And here are my acknowledgments.

The DARPA System F6 Program supported this work. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of DARPA.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Motivation

Cyber-Physical Systems (CPS) define classes of systems in which embedded computers run sensing and actuation control software to interact closely with a physical system in a physical environment. In these systems, the physical environment in which the computer exists is tightly coupled with the computer and its software since the computer controls some aspect of the physical system. Because of this tightly coupled nature, the software not only affects the computer and its surrounding environment, but also is affected by the surrounding environment. Many different types of systems fit this CPS description: e.g. autonomous vehicles including Unmanned Aerial Vehicles, Unmanned Underwater Vehicles, autonomous cars, and embedded or wireless sensors/actuators. As an example, consider a satellite and its control software. The control software reads the sensor data from the satellite's sensors to determine the current state vector of the satellite. Using this state information, the software's feedback control loop governing the orbital position of the satellite may determine that a thruster on the satellite must be activated to correct the satellite's orbit, e.g. orbit degradation caused by atmospheric friction or gravitational perturbations. The satellite has very limited resources, both with respect to physical resources like propellant, as well as computing resources, like power and processor time. Because of these resource constraints, if the software component involved with determining the state of the system does not meet its timing deadlines and instead calculates the satellite state too late (e.g. high computational load causing the Kalman-filter state estimation to miss its deadline), the satellite may not have enough propellant to properly correct its trajectory. Similarly, the software component involved with activating the satellite's thruster must meet its deadlines otherwise the satellite will not achieve the proper orbit.

As these types of CPS are being scaled-up, they are becoming more distributed in nature. The systems mentioned above could scale up to unmanned swarms of search and rescue drones, for instance, or large sensor/actuator networks for power distribution and control. The scaling of these systems is possible through the use of the system's network, which facilitates the communications between the nodes of the system and in doing so acts as the critical backbone allowing distributed systems to function. We continue our satellite example, whose scaling up could lead to a cluster of satellites cooperating, communicating and running distributed applications in service of the mission goals. Because of this trend towards cooperating distribution of system resources, the network facilitating the cooperation and communications becomes a critical resource to the system. Whereas the single satellite's internal communications bus (direct physical connection system internal to a single satellite which allows sensing and actuation controlled by the computer) was ignored in the previous example, the wireless communications network enabling the satellite cluster cannot be ignored when analyzing the properties of the system. Because the satellites are expensive to deploy, impossible to repair, and must last for a long time to satisfy both budgetary constraints and mission goals, the application developers and system integrators for the cluster must ensure that the software on the cluster does not compromise its ability to meet the mission goals. For instance, the same orbit maintenance described previously now necessitates the use of the cluster's network. For a satellite to activate its thruster to maintain or modify its orbit, it must first ensure that such an action will not cause a collision with another of the satellites in the cluster. Therefore, every satellite must know the state of every satellite in the system, and any thruster activation must be a coordinated action to ensure the safety and continued operation of the cluster. All of this state distribution and coordination occurs over the wireless network between the satellites, which (1) has limited resources, (2) is shared between all applications on all the satellites, and (3) varies as a function of time throughout the orbits of the satellites according to the orbital mechanics defining the system. The third point is especially important, since it highlights how the

physics of the system directly and drastically affects system resources and therefore system performance and stability. Again, we must ensure the timing properties of the sensing and actuation are met, except now those timing properties are directly related to the network resources, e.g. the transmission and buffering delay in the network links, the bandwidth of the links, and the buffer space available to the applications on each satellite.

The link between these network resources and the system provides a contract which specifies the service that users require from the system and that the system can provide to each user. The quality of this service as seen by the users of the system is defined as the Quality of Service (QoS) of the network and is the overall performance of the network as seen by its users. The specific aspects of QoS which we focus on are the network bandwidth, transmission and buffering delay, and availability of the network resources. For critical systems such as those described above which may be quite difficult to repair or replace, such requirements must be analyzed at design-time and verified to ensure that they are met. For any distributed CPS, the network performance of the system is affected by the physical environment of the system. For systems whose physical network layer is comprised of wired connections, this effect may stem from the temporal properties of the control systems and their periodic or sporadic network load. For systems whose physical network layer is made up of wireless connections, the physical environment has an even larger effect on the network resources and availability. Environmental interference or obstruction leading to multi-path self-interference or signal degradation can combine with the distance-based signal-to-noise ratio loss due to the nature of wireless media. Because the network performance of such a system is so tightly coupled with the physics governing the system, the physical dynamics must be taken into account when predicting the run-time characteristics of the network. Additionally, such resource constrained systems which are expensive to develop and deploy must maximize their return on investment through the

hosting of payload applications (e.g. for scientific research), while ensuring that the resource requirements are not exceeded. This design-time analysis of time-varying resources and their constraints is paramount to ensuring a stable system.

Incorporating the physical dynamics into the model of the system network resources addresses only half of the problem, however. To facilitate accurate, meaningful resource constraint analysis, the application developers can model and describe the resource and timing constraints of their applications. As stated above, many of these systems have long-term missions, for which simple, static minimum/maximum resource and timing requirements lead to inefficient, underutilized, over-specified systems. To increase the fidelity of the application resource utilization model with respect to the actual application's resource usage, the time-varying nature of the application's network utilization should be modeled. In this way, tighter bounds on performance characteristics and resource utilization can be achieved. Tighter bounds on application performance and resource utilization allow system integrators to increase overall system resource utilization to maximize the mission-specific or scientific return of the system while still ensuring all applications receive their required services.

In addition to the design-time modeling and analysis which facilitates the calculation of performance guarantees about such critical CPS, the run-time systems require monitoring and management of resources and their utilization to prevent faulty or malicious applications from causing resource over-utilization and possibly making the system unstable or completely bringing the system down. Often this resource management is simply enforcing a static cap on resource utilization for each application. For such trivial resource management, often the operating system or other platform infrastructure is used to enforce these bounds on the applications' resources, e.g. open file descriptor limits or maximum buffer size limits being enforced in the Linux kernel. However, higher fidelity design-time models which more precisely capture the behavior and resource requirements of the applications can allow more sophisticated, time-varying resource monitoring and management.

Another type of adaptive resource management falls under the class of self-adaptive systems, which are capable of self-management at run-time. Using recent developments in autonomic computing, systems can use the sensors at their disposal to monitor their available resources as well as their environment, estimate the current state of the system, and use the available system actions to transition into a new state. A relevant example for such an adaptive system would be to eschew the design-time network modeling and analysis of what at run-time would be a relatively static system in favor of an adaptive network which manages the network resources for the applications based on the available resources the system has. Such a design has the benefits of possibly better utilization of system resources and better resilience to unplanned or unforeseen system events or states, but has the drawback of difficult design-time analysis. Currently, analyzing these adaptive systems at design time to derive guarantees about system behavior, resource availability, or performance is quite difficult and in many cases infeasible.

1.2 Challenges

The systems described above face many challenges for network performance prediction, as might be required by mission- or safety-critical application developers. Furthermore, an application which consumes more resources than specified at design-time, either through malicious or faulty code, can send these CPS into an unstable state by starving critical control processes of resources. Such resource over-utilization should be prevented at run-time to ensure such unstable states are avoided. In this section we outline the main challenges facing application developers and system integrators pertaining to network performance prediction and management and we separate these two classes of challenges into Design-Time Analysis challenges and Run-Time Management challenges.

1.2.1 Design-Time Network Performance Analysis of Distributed CPS Applications

A principal challenge of system design is the performance analysis of a system, its resources, and its applications at design-time. Such analysis and prediction is critical for remotely managed systems and allows system integrators to provide guarantees to application developers about the services provided by the system. However, for complex distributed cyber-physical systems such design-time analysis is challenging. Such analysis may require capturing the behavior of the system and its applications in models that can then be composed and analyzed. Ensuring that the models properly capture the relevant characteristics of the run-time system is a challenge by itself, and is compounded by the challenge of composing the models for analysis. Such challenges for design-time network performance analysis are

- Modeling the interaction of the system with the physical world is difficult, esp. with respect to how the interaction directly or indirectly affects system resources and performance.
- Application network utilization models can be imprecise and difficult to derive without a running system
- Developing distributed applications for such systems is difficult, and should be done in a way that is amenable to modeling, analysis, and verification.
- Infrastructural code which handles low level system functions or network communications complicates application development
- Network resources are becoming more critical to distributed CPS, but design-time analysis of network resource utilization and performance is difficult
- For resource constrained systems, no buffer-space or processor time should be wasted, but without accurate and precise design-time analysis, it is difficult to properly specify network resource requirements.

- For application/system data flows in the network which require tight and/or real-time guarantees on temporal properties, design-time analysis is critical.

- Most systems have some form of routing, either static or dynamic; dynamic routing is difficult to analyze for precise performance prediction

1.2.2 Run-Time Network Resource Monitoring and Management

Given specifications for system network resources and application network resource requirements, the system must ensure that no application either purposefully or inadvertently exceeds its allowed resource limits and starves other applications or critical system processes of those precious resources. Such resource management is crucial for ensuring system stability and proper service quality to applications and end-users. For systems with highly time-varying application load, system resource availability, or both, static limits under-utilize the system's resources. For such systems, higher fidelity resource management is needed to maximize the utilization of the system's resources. Further, these higher fidelity system and application network resource models pave the way for more accurate and robust failure or attack detection which in turn can provide higher system stability. Challenges towards the development of such run-time network resource management are

- Available network resources at run-time should not be wasted if applications can use them, but allowing run-time management is difficult because the behavior is difficult to analyze at design-time for performance analysis and prediction.

- Applications which attempt (either due to faults or attacks) to use more network resources than they originally specified should be detected and mitigated; the detection of coordinated attacks, e.g. distributed denial of service (DDoS), requires more sophisticated detection and mitigation techniques

- Systems are becoming more adaptive in nature and reacting to events at run-time

(essentially data-dependent traffic); this adaptability is hard to provide performance metrics or guarantees for

1.3 Organization

The rest of this proposal is organized as follows

- Chapter II describes the related work in network analysis and management of distributed applications
- Chapter III describes design-time network performance analysis and prediction for CPS applications, including completed and proposed work
- Chapter IV describes run-time network performance monitoring and management for CPS applications, including completed and proposed work
- Chapter V concludes the proposal and provides a tentative time-line for the proposed research
- Chapter VI lists the publications so far

CHAPTER II

RELATED WORK

2.1 Part 1: Design-Time Network Analysis and Performance Prediction

Networking systems have been developed for over half a century and the analysis of processing networks and communications networks began even earlier. As computing power has increased, the field of network performance analysis at design-time has evolved into two main paradigms: (1) network performance testing of the applications and system to be deployed to determine performance and pitfalls, and (2) analytical models and techniques to provide application network performance guarantees based on those models. The first paradigm generally involves either arbitrarily precise network simulation, or network emulation, or sub-scale experiments on the actual system. The second paradigm focuses on formal models and methods for composing and analyzing those models to derive performance predictions.

2.1.1 Performance Analysis Through Network Simulation/Emulation

Since computing networks are so prevalent, many tools exist to analyze system network behavior, either through simulation or mathematical analysis, which both attempt to determine one or more system properties based on one or more models of the system. One method of system simulation is discrete event simulation[30], in which all relevant events in the system are captured in the model and stepped through sequentially with the state of the model changing only at the simulated time steps. The resources of the system (e.g. buffer space) are simulated together with the entities in the system (e.g. the bits of the network traffic) through operations on the entities as they traverse the model and its resources. OMNET++[32] is a discrete event network simulator which simulates the network traffic as it passes through the network layers. The INETMANET framework, built on top

of OMNET++, supports the simulation of network traffic over dynamic wireless links for gathering performance data about applications on the network.

NS-2[25] is a widely-used *single-threaded* discrete event simulator which allows both the simulation and emulation of wireless networks. Because of performance and scalability issues, however, the simulator is not well suited for scaling to large network simulation/emulation. Furthermore, NS-2 has simulation accuracy issues (e.g. altering event ordering or timing) which plague any simulator used for emulation (i.e. connecting a simulator to a system to emulate the subsystem it is simulating). [16] gives a good study of the accuracy of NS-2 simulation with a testbed and finds that for constant bit-rate (CBR) traffic the simulation is accurate with respect to the behavior of the real system testbed, but for other types of traffic (e.g. FTP traffic), the simulation did not accurately model the dynamic behavior of FTP traffic.

Despite the wide-spread use of these simulation toolsuites, it is clear that they are not a viable candidate for providing both accurate and precise design-time guarantees about network performance and resource utilization.

Instead of simulating the network/stack, another option is to directly emulate the network by shaping the traffic between the actual nodes of the system to directly apply the appropriate delay and enforce the proper bandwidth on each link of the network. Often this is done through the use of flow control tools either on routing node(s) or on capable network infrastructure devices, e.g. a smart switch. Dummynet[28][6] is a tool for network emulation when used on routing nodes in a system, utilizing the underlying network traffic shaping and policing tools available in Linux. Dummynet allows the configuration of routing tables, packet drop rates, link bandwidths, and link delays to conform the traffic passing through it to the supplied network configuration. OpenFlow[7] is an alternative for network emulation which instead uses a compatible smart switch to shape the network traffic and enforce the proper network topology and characteristics for all traffic in the network at a lower network layer without requiring the use of a separate traffic shaping node.

For the types of systems we have described in Chapter I, typically these types of simulation and testbed emulation are used to analyze the performance of the applications and the system. Unfortunately simulation and emulation based performance analysis techniques are unable to provide the guarantees required by application developers and system integrators.

2.1.2 Analytical Approaches to Network Analysis

2.1.2.1 Queuing Theory

Queuing Theory[19][15] is a probabilistic approach to the analysis of processing or communications networks, and has been applied to many types of systems including telecommunications, processing, and distribution systems. A queuing system can be described using notation of the form $A/B/S/\Delta/E$, introduced by [19], with the semantics:

- A: Type of arrival process, e.g. M for Poisson Arrival Process
- B: Request service time statistics, e.g. D for Deterministic service time
- S: Number of servers
- Δ: Queue length
- E: Number of producers

Queuing Theory allows the analysis of the mean number of requests (N) in the queue and the mean buffering delay (T) experienced by objects traversing the queue. Little's Theorem provides the relation between the two : $N = \lambda T$[23], where λ is the mean arrival rate into the queue. However, this theorem assumes (1) that the service policy is independent of service time and (2) the service policy is work conserving. Assumption (1) may be violated for policy-based routing and servicing which tries to provide guaranteed QoS to applications, and assumption (2) is violated by wireless networks, in which nodes with very limited connectivity or dropouts in connectivity are not able to service the data in

the buffers despite the existence of the data in the buffers and applications continuing to produce data.

For the types of systems we have described in Chapter I, probabilistic analysis techniques like Queuing Theory make providing the requisite performance and resource guarantees difficult or impossible. Because of the need for these strict guarantees, other deterministic formal models for the analysis of communications and processing systems have been developed.

2.1.2.2 Network Calculus and Min-Plus Calculus

Network Calculus[9][8][21] is a theory for deterministic queuing systems which provides the ability to determine worst-case buffer requirements and application buffering delay at design-time by applying the techniques of (min,+) calculus to queuing theory. We will describe the foundation of (min,+) calculus before covering the techniques of Network Calculus. [21], Chapter 3, gives an excellent overview of both min-plus and max-plus calculus, on which Network Calculus is based. An abbreviated explanation of the concepts of these two related dioids (additive inverses need not exist) follows.

Min-plus calculus, $(\mathbb{R} \cup \{+\infty\}, \wedge, +)$, deals with *wide-sense increasing functions* :

$$\mathcal{F} = \{f : \mathbb{R}^+ \to \mathbb{R}^+, \forall s \leq t : f(s) \leq f(t), f(0) = 0\} \tag{1}$$

which represent functions whose slopes are always ≥ 0. Intuitively this makes sense for modeling network traffic, as data can only ever by sent or not sent by the network, therefore the cumulative amount of data sent by the network as a function of time can only ever increase or stagnate. A wide-sense increasing function can further be classified as a sub-additive function if

$$\forall s,t : f(s+t) \leq f(s) + f(t) \tag{2}$$

Note that if a function is concave with $f(0) = 0$, it is sub-additive, e.g. $y = \sqrt{x}$. Sub-additivity of functions is required to be able to define meaningful constraints for network calculus, though realistically modeled systems (in Network Calculus) will always have sub-additive functions to describe their network characteristics (e.g. data serviced or data produced). This sub-additivity comes from the semantics of the modeling; since the models describe maximum data production or minimum service as functions of *time-windows*, maximum data production over a longer time window must inherently encompass the maximum data production of shorter time-windows. Some examples of wide-sense increasing functions which are of use in Network Calculus are shown in Figure 1.

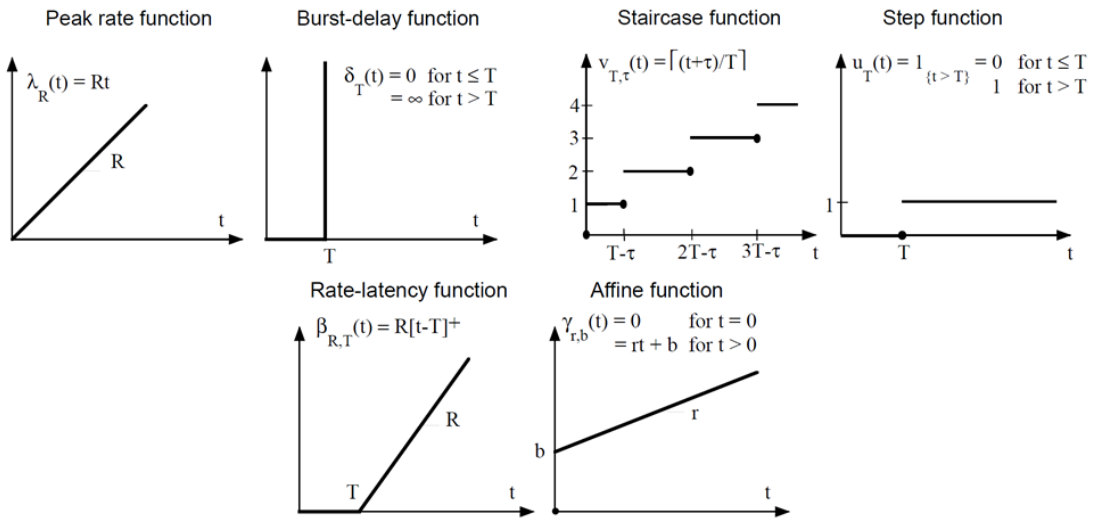

Figure 1: Example wide sense increasing functions, reprinted from [21].

The main operations of min-plus calculus are the convolution and deconvolution operations, which act on sub-additive functions. Convolution is a function of the form:

$$(f \otimes g)(t) \equiv inf_{\{0 \leq s \leq t\}}\{f(t-s) + g(s)\} \tag{3}$$

Note that if the functions f, g are concave, this convolution simplifies into the computation

of the minimum:

$$(f \otimes g)(t) = min(f, g) \tag{4}$$

Convolution in min-plus calculus has the properties of

1. closure: $(f \otimes g)(t) \in \mathcal{F}$,
2. Associativity,
3. Commutativity, and
4. Distributivity

Similarly, deconvolution is a function of the form:

$$(f \oslash g)(t) \equiv sup_{\{0 \leq u\}}\{f(t+u) - g(u)\} \tag{5}$$

Note that $\oslash$ is not closed in $\mathcal{F}$ because $(f \oslash g)(t)$ is not necessarily 0 for $t \leq 0$.

Network Calculus focuses on abstracting the network traffic and the computing nodes as *arrival curves* and traffic shaping *service curves*. The arrival curves and service curves model the amount of data generated or serviced as functions of time window size and are bounded by maximum and minimum arrival and service curves. By abstracting the network flows and traffic shapers as arrival curves and service curves, respectively, (min,+) calculus can be used to compose models of system behavior and calculate performance characteristics of the application and the network.

Given an arrival function $R(t)$ for the data flow describing the number of bits seen on the flow during the time interval $[0,t)$, the arrival curve α constrains the flow if and only if

$$\forall s \leq t : R(t) - R(s) \leq \alpha(t-s) \tag{6}$$

This relation is shown in Figure 2. Intuitively the arrival curve representation transforms the

data production from a function of time, described by $R(t)$, into a function of time-interval, described by $\alpha(t)$, for which $R \leq R \otimes \alpha$.

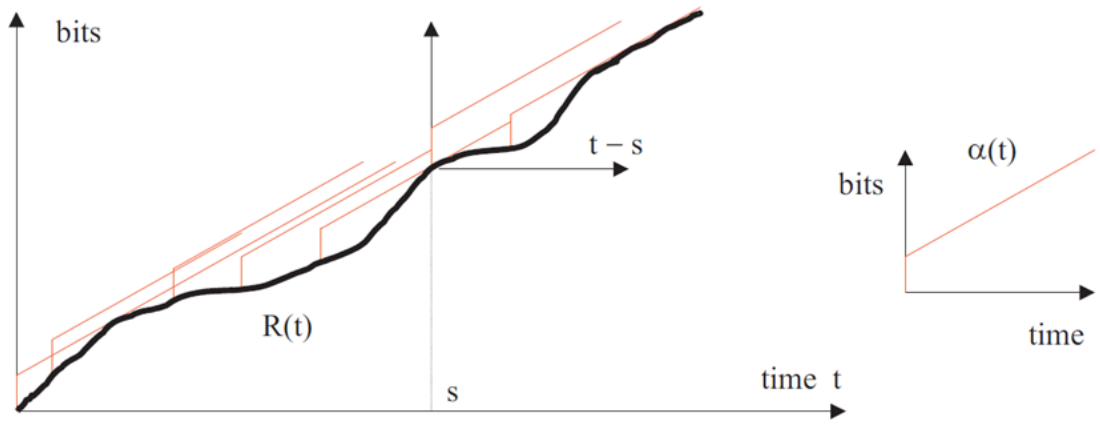

Figure 2: Illustrative example representing maximum arrival curves ($\alpha(t)$) for data flows ($R(t)$), reprinted from [21].

Similarly, service curves transform the output data flow $R^*(t)$ into a minimum service curve β according to the relation:

$$R(t) - R^*(t_0) \leq \beta(t - t_0), \forall t \geq 0 \; \exists \; t_0 \geq 0, t_0 \leq t \tag{7}$$

or more compactly $R^* \geq R \otimes \beta$. This relation is shown in Figure 3.

From the input arrival curve α into a node providing service curve β, we can use Network Calculus to compute the output flow from the node and a few performance bounds governing the buffering delay and buffer requirements for the node. The output flow from the node is constrained by the arrival curve $\alpha^* = \alpha \oslash \beta$. Given the arrival curve and service curve for a node or system, we can compute the backlog and delay bounds, see Figure 4; the backlog bound is given by:

$$R(t) - R^*(t) \leq sup_{\{s \geq 0\}} \{\alpha(s) - \beta(s)\} \tag{8}$$

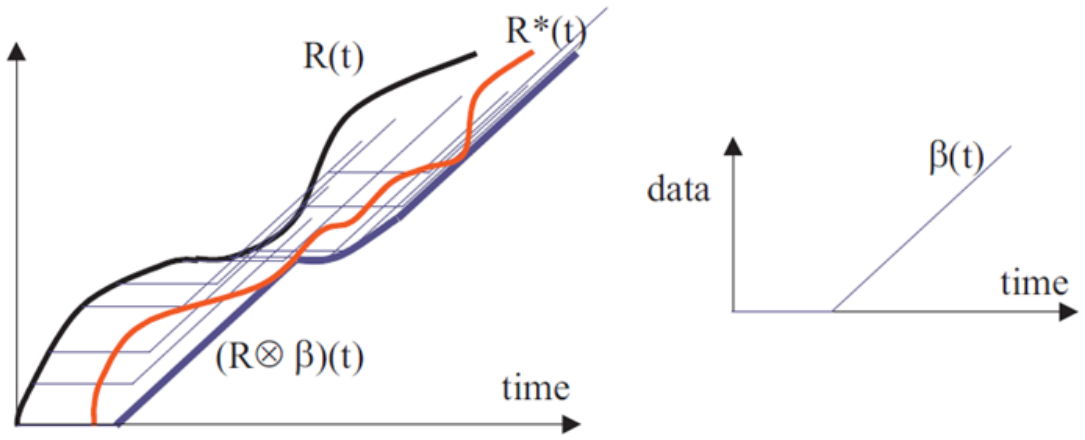

Figure 3: Illustrative example representing minimum service curves ($\beta(t)$) for output data flows ($R^*(t)$), reprinted from [21].

and the delay bound is given by:

$$h(\alpha,\beta) = sup_{\{s\geq 0\}}[inf\{T : T \geq 0 \; and \; \alpha(s) \leq \beta(s+T)\}] \tag{9}$$

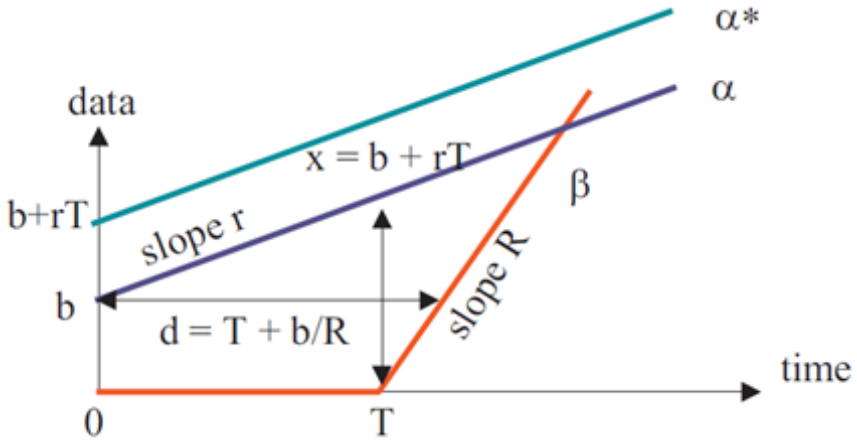

Figure 4: Illustrative example representing the backlog and delay bounds calculated from input arrival curves and node service curves, reprinted from [21].

These bounds provide the requisite information needed to make design-time guarantees

about *worst-case* application performance on the network, given that both the application traffic profile and the system's network performance are deterministic.

To enable compositional system analysis, Network Calculus allows for the concatenation of nodes, Figure 5, such that a flow traversing nodes N_1 and N_2 in sequence, where each node provides FIFO service curve $\beta_{i=1,2}$, the concatenation of the two nodes offers a service curve $\beta_1 \otimes \beta_2$ to the flow. A major advantage of this approach is the ability to "Pay Bursts Only Once" (PBOO), which is the property that the delay and buffer bounds are tighter when derived from the concatenation of the system than they would have been if they were calculated iteratively. Again, note that this advantage is not applicable to non-FIFO systems[21].

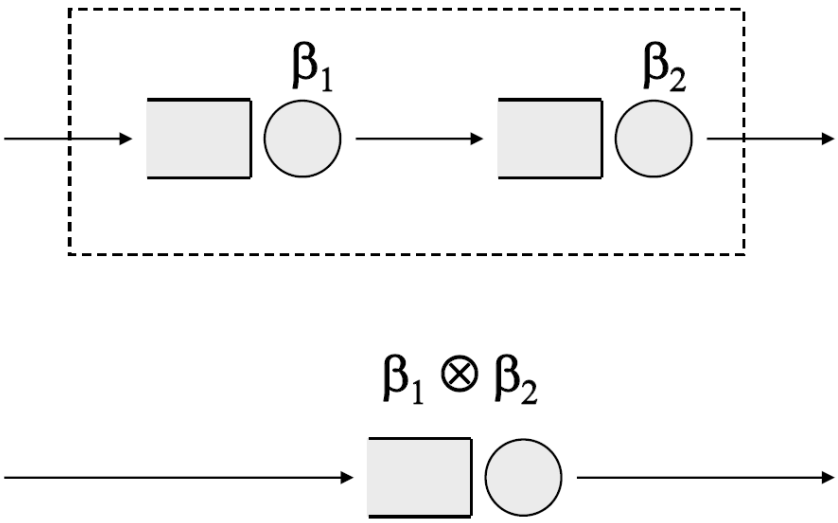

Figure 5: Illustrative example representing the concatenation of two nodes providing separate services into a single node providing an aggregate service.

However, the performance bounds calculated by Network Calculus are still worst-case performance based. For instance, there is a temporal disconnect between the arrival/service curves and the actual performance of the application or the system. This disconnect

leads to analysis results that may still over-approximate the required buffer size or application delay on the network. The cause of this over-approximation comes from the use of *time windows*. Because Network Calculus is focused on maximum data produced and minimum data serviced as functions of time window size, the time-varying nature of the data production or service is lost. Despite an application producing a Bulk Data Transfer (BDT) during a period of high network resource availability, Network Calculus compares that BDT to all windows of time throughout the service time of the system. As such, an expected drop in service during a different period of time will inadvertently negatively affect the application's predicted performance as analyzed by Network Calculus.

2.1.2.3 Real-Time Calculus

Real-Time Calculus[31] builds off Network Calculus, Max-Plus Linear System Theory, and real-time scheduling to analyze systems which provide computational or communications services. Unlike Network Calculus, Real-Time Calculus (RTC) is designed to analyze real-time scheduling and priority assignment in task service systems. The use of (max,+)-calculus in RTC allows specification and analysis not of only the arrival and service curves described above for Network Calculus, but of upper and lower arrival curves ($\alpha^u(\Delta)$ and $\alpha^l(\Delta)$) and upper and lower service curves ($\beta^u(\Delta)$ and $\beta^l(\Delta)$). These curves represent the miniumum and maximum computation requested and computation serviced, respectively. An overview of RTC is given in Figure 6.

RTC allows for the analysis of task scheduling systems by computing the request curve for a task model which is represented as a directed acyclic graph (DAG), the task graph $G(T)$. The graph's vertices represent subtasks and each have their own associated required computation time $e(u)$, and relative deadline $d(u)$ specifying that the task must be completed $d(u)$ units of time after its triggering. Two vertices in $G(T)$ may be connected by a directed edge (u,v) which has an associated parameter $p(u,v)$ which specifies the minimum time that must elapse after the triggering of u before v can be triggered. RTC develops

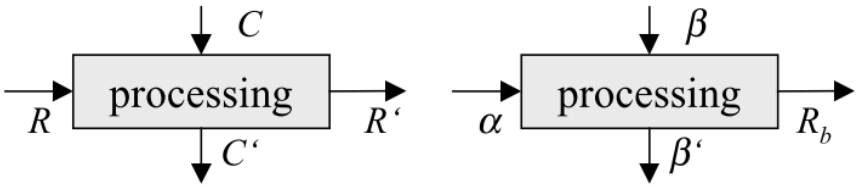

Figure 6: Overview of Real-Time Calculus' request, computation, and capacity models. $R(t)$ is the request function that represents the amount of computation that has been requested up to time t, with associated minimum request curve, α. $R'(t)$ is the total amount of computation delivered up to time t, with associated delivered computation bound $R_b(t)$. C and C' are the capacity function and remaining capacity functions which describe the total processing capacity under full load and the remaining processing capacity, respectively. C and C' are bounded by the delivery curve β and the remaining delivery curve β'.

from this specification the minimum computation request curve α_r and the maximum computation demand curve α_d. Finally, the schedulability of a task T_i is determined by the relation:

$$\beta'(\Delta) \geq \alpha_d^i(\Delta) \quad \forall \Delta \tag{10}$$

which, if satisfied, guarantees that task T_i will meet all of its deadlines for a static priority scheduler where tasks are ordered with decreasing priority. Note that the remaining delivery curve $\beta'(\Delta)$ is the capacity offered to task T_i after all tasks $T_{1 \leq j < i}$ have been processed. Similarly to Network Calculus, RTC provides analytical techniques for the computation of performance metrics such as computation backlog bounds:

$$\text{backlog} \leq sup_{\{t \geq 0\}} \{\alpha^u(t) - \beta^l(t)\} \tag{11}$$

which is equivalent to the network backlog bound derived in Network Calculus.

[14] compares different analytical methods for network performance analysis, namely Real-Time Calculus (RTC), probabilistic queuing models, parallel computation models, and protocol offload models. The authors explain the current state of system evaluation, which is based predominantly on quantitative evaluation through simulation, but make the

point that such simulation techniques should be used sparingly since *only a finite state of initial states, environment behaviors, and execution traces can be considered by system simulators*. The system which the authors model for their comparison is the case of Network Interface Cards (NICs) connected to a Local Area Network (LAN), for which they derive analytical bounds on the buffer requirements as they are affected by the input/output (I/O) subsystems, the network traffic, the NIC itself, and the memory controllers. They point out that most researchers are still using Queuing Theory, in stochastic scenarios where the network traffic is modeled as random distributions of data. Because RTC allows more precise descriptions of application traffic, it can be more beneficial for providing analysis of buffer requirements and delay experienced in the system. RTC's ability to allow such specifications comes from its roots in Network Calculus and Max-Plus Linear System theory.

2.1.2.4 Stochastic Network Calculus

These deterministic constraints can be relaxed so that the deterministic arrival and service curves are instead replaced by stochastic processes, causing the bounds on the performance to be probabilistic as well[5]. As described previously, these probabilistic performance bounds may not be precise enough to provide the types of guarantees required by certain classes of mission- or safety-critical systems.

[33] provides a good description, system model, and analysis for stochastic Network Calculus applied to wireless networks. In their work, they show the ability of network calculus to remove the need to make as many assumptions about the arrival or service processes (*e.g.* exponential service distribution) to allow general arrival and service processes. They apply stochastic network calculus to analyze backlog and delay bounds in 802.11 based multi-access systems. They formally derive bounds for the backlog and delay in the network and then compare these analytical results to bounds generated from network simulation using ns-2. From this comparison, they conclude that the derived bounds are too

loose and in fact get looser the closer the system gets to saturation. They further conclude that this looseness is a direct result of stochastic network calculus itself, and claim that it requires further improvements. It is important to remember what was stated previously by [16]: the simulation does not accurately model the dynamic behavior of real traffic, so the results from [33] may too be inaccurate.

Because Network Calculus deals with either deterministic worst-case application performance on a static network or stochastic application performance on a dynamic network, system designers and application designers under-utilize the network resources of systems which require strict design-time guarantees about application performance.

2.1.2.5 Extensions to Network Calculus

When analyzing any complex system, the fidelity of the analysis results with respect to the actual system relies heavily on the level of detail of the models of the system's components and subsystems. [17] covered the effects that different levels of detail have on analysis complexity and accuracy. Importantly, they point out the requirement to not only be correct, but also be applicable, *i.e.* analysis results should be both accurate with respect to the system being modeled, but should also be relevant for the analysis and development of real systems. Additionally, they point out that not all systems require highly detailed modeling for the analysis results to be correct, since some systems and applications are insensitive to lower level details.

There are many efforts to make analytical techniques more representative of actual systems in order to increase the fidelity of the analysis results with respect to the run-time system. The researchers in [22] recognize the need to analyze not only the overall throughput of a network, but also the end-to-end delay experienced by information flows in the network. Furthermore, they derive an analytical model of Wireless Network Coding[18], a technique for combining packets together for improving network throughput in wireless networks using broadcast techniques. They show that by developing a model of the way

the MAC layer works in the network and how the information flows are combined and disseminated, they can get tighter performance bounds and even derive methods for increasing performance in the network by altering the scheduling parameters of the packet flows. Analyzing multiple performance parameters, in this case the network throughput and end-to-end delay, is a key element for analyzing and providing quality of service (QoS) to applications.

The authors in [3] also incorporate more precise models of the network to derive tighter performance bounds using Network Calculus. They show that by modeling the packetization that occurs in the network using a *packet operator* to transform arrival flows into packet flows, they can analytically derive tighter service curves than would be found from traditional Network Calculus. Clearly, there exists a desire from application developers and system integrators to derive both accurate and precise design-time performance parameters for the system and its applications.

Similarly, in [12], the authors describe how to accurately model the SpaceWire network standard which has been developed for satellites in the European Space Agency (ESA). Their network must be shared by both real-time (critical) and non real-time (non-critical) traffic, but the system developers require design-time guarantees about the temporal characteristics of all critical/real-time messages on the network. Their work focuses on accurately representing the SpaceWire network, its (static) routing protocol, and the service profiles of its routers including the aspects of their flow control algorithms. Building on previous work, they explain the need, for resource-constrained real-time systems, to accurately model the network traffic in order to derive a model of the network which is not too pessimistic. They derive accurate Network Calculus/Real-Time Calculus (RTC) based models of the wormhole switches present in the network and show the fidelity of their analytical tools compared with the industrial simulation tools developed for SpaceWire networks. Using a Network Calculus-based model, they are able to achieve analytical results that are the

same order of magnitude as the simulation results for the critical traffic delay characteristics, but are less precise for the non-critical traffic in the network. One important point they make that extends to all types of systems when comparing analysis and simulation techniques is this: *worst-case delays can be extremely rare events which are hard to observe or recreate in simulations, but can be derived from analytical results.*[12]

Another approach to increasing the fidelity of the analysis is to model the Time Division Multiple Access (TDMA) medium channel access protocol using Network Calculus to derive performance metrics[29]. TDMA service curves are modeled such that the medium's transmit capacity is available to the node only during the node's designated slot. During all other slots of the TDMA period, the medium's capacity is unavailable to the node and therefore the transmit capability of the node is zero. As such, simple TDMA service curves can be described using simply a slot length, a slot bandwidth, and a TDMA period.

[20] analyzes the performance of TDMA with respect to the queue size for different probabilistic traffic models, and shows how the G/D/1 model with application-based probability distributions can be used to generate closed-form solutions for analyzing arbitrary traffic on a TDMA network.

Another aspect of system design which has been gaining momentum is the development of self-adaptive systems which provide "self-*" properties such as self-management. These types of systems are typically not used in CPS control applications or other systems which require real-time guarantees about timing or resource properties of the system. The main reason for their absence from these types of systems and applications is the lack of available, accurate modeling and analysis techniques which properly capture the behavior of the applications in a way that allows the derivation of performance guarantees. The authors in [10] describe both the need for this type of analysis for these systems and describe the overview of how the analysis would work, based on concepts from Network Calculus. Their main point is that currently such types of analysis tools do not exist for these systems, which makes developing the systems difficult with respect to these types of design

parameters. They posit that developing a formalized standardization for the self-adaptive behavior, which they present as a state-space with available control actions based on the sensor data in the system.

2.2 Part 2: Run-Time Network Monitoring and Management

In addition to design-time modeling and analysis, CPS system designers and integrators must ensure system stability during run-time by enforcing resource limitations on the applications to ensure no faulty or malicious code starves the system or other applications of network resources. Such enforcement is the management of the network resource for the system. Many different approaches exist to handle this type of management, generally falling into one of two categories: (1) static management or (2) dynamic management. Static management of system resources is based around enforcement of resource allocations which were decided at design-time or deployment time. Such management generally is associated with high-criticality systems which must be guaranteed. Dynamic management of resources entails updating the resource allotments of each application based on currently available system resources and application load, and generally is in the class of adaptive management or adaptive systems (also called autonomic systems). For this paper, we will address only static management of resources.

Static management of network resources generally, but not necessarily, means applications are given a fixed set of resources for the lifetime of the system. The part of the system which enforces these resource allotments however, may vary depending on the design of the system. The enforcement may happen in the network layer, in the operating system kernel, or in some cases in the middleware facilitating the network communications for the applications. We deem any enforcement happening in the kernel or in a lower layer to be *infrastructural management* (since all applications on the system must use this infrastructure and are therefore managed by it). We deem any resource management happening

between the kernel and the application as *middleware management*, since different applications deployed on the system may use different middleware stacks and therefore may be managed differently.

2.2.1 Infrastructural Approaches for Network Management

[15], Chapter 3 has a good overview of both IntServ and DiffServ. DiffServ, for Differentiated Services, is designed for the provisioning of network resources to provide Quality of Service (QoS) to applications on the network but is unable to provide strict real-time guarantees about packet loss, delay, and bandwidth availability. Instead, DiffServ was designed to scale well for large systems while still providing probabilistic guarantees. IntServ, for Integrated Services, was designed to provide strict real-time guarantees about the QoS experienced by a flow on the network. Unlike DiffServ, which does not maintain any state information in the routers along network flow paths, IntServ uses a resource reservation protocol (RSVP) with explicit setup of flows for deterministically allocating bandwidth and buffer space for a flow in each router along the flow's path. While such an explicit out-of-band QoS reservation protocol enables similarly explicit resource availability and performance guarantees, the tradeoff comes in the ability of the system to scale to many nodes and many flows. DiffServ's scalability comes from both the lack of explicitly maintaining per-flow state in the routers, by assigning traffic to a set of predefined classes, as well as using QoS assignment mechanisms which are built into the flow's messages, e.g. the DiffServ Code Point (DSCP) built into the Type of Service (ToS) byte in IPv4 headers and the Traffic Class byte of IPv6 headers.

Both IntServ and DiffServ were originally designed for wired networks, but [24] has worked on the required modifications to make them suitable for wireless networks, which have network connectivity and link characteristics which have more variance as a function of time. The combination of low bandwidth, high loss, and node mobility require extensions to the QoS parameters and control options available to the application provided by

the QoS infrastructures. One such proposed extension is the loss profiles, which govern whether an application prefers dropping data in a bursty manner (as might be preferred by audio applications) versus a distributed manner (as might be preferred by video applications). Similarly, since link bandwidth is typically much lower than in wired networks, IntServ/RSVP's refresh messages (used to determine network changes) should be sent with a lower frequency to provide as much network bandwidth as possible to application traffic. In the same way, DiffServ requires modifications to support signaling information about link state and node location to overcome DiffServ's static provisioning scheme in the adaptation from wired to wireless networks.

A system's network infrastructure may provide multiple different QoS provisioning implementations, such as both DiffServ and IntServ. In this case, the applications can select which QoS provisioning to use. Similarly, large networks may be grouped into subnets which each internally use different QoS provisioning schemes. The boundaries between these subnets requires QoS mapping for flows which cross these boundaries. Such mapping between QoS implementations and configurations is complex and makes providing guarantees about QoS for large complex networks challenging.

Because both IntServ and DiffServ were designed for providing QoS to generic traffic for large networks including the internet, they were not designed to be able to provide performance guarantees to application developers. As such, their design and implementation function more coherently in a system which has unknown applications and application load. However, the classes of systems we focus on require more precise guarantees about performance and have the benefit of more precise design-time knowledge of applications and application load on the system.

Flexible QoS Model for Mobile Ad-hoc Networks (MANETs), FQMM[34], attempts to address the issue of run-time QoS management and adaptation to changing environmental conditions affecting the network. Recognizing that both environmental and application

behavior need to be taken into account for QoS management, they argue that two methods for providing QoS in the internet, IntServ[4] and DiffServ[2], are not sufficient for dynamic mobile networks. While IntServ's scalability problem will not affect dynamic mobile networks in the near future, they argue that the connection maintenance required by the Resource ReSerVation Protocol (RSVP) renders IntServ impractical. DiffServ, on the other hand, might be able to provide long-term QoS to applications under the varying network conditions, but is not feasibly able to provide the kind of short-term QoS required by real-time applications. Furthermore, DiffServ does not handle node mobility and external disturbances from the environment well as it was originally designed for relatively fixed (topologically) networks.

To combat the issues in both IntServ and DiffServ, FQMM is designed to handle QoS for MANETs. FQMM focuses on allowing for fine-grained provisioning of node resources and allowing node mobility through dynamically reassigning the roles of each of the nodes in the network. The provisioning of the resources for flows in the network borrows ideas from both IntServ and DiffServ by combining the per-flow granularity of IntServ for high-priority flows while lower-priority flows are provisioned on a class basis as in DiffServ. This differentiation between traffic classes and priority flows better utilizes the system resources to achieve the necessary performance for high priority flows which may need real-time performance. To provide traffic shaping they constrain flows or classes to traffic profiles. To combat the time-varying nature of the network, they instead define these traffic profiles as a percentage of the available network bandwidth. This type of percentage-based flow constraint limits the adaptability of the network traffic however, as certain higher-priority real-time flows may have a minimum amount of bandwidth required that cannot be met with a percentile constraint on effective link bandwidth. FQMM also addresses routing control to provide better run-time QoS to applications on the system.

2.2.2 Middleware Based Approaches for Network Management

For system and application level adaptation to changing system resources, two main approaches, namely fixed reservation of flows and run-time adaptation, provide benefits for performance or resilience. These two approaches cannot be used alone however, as fixed reservation of flows based on design-time network analysis causes low resource utilization and run-time adaptation is generally not prepared for excessive congestion or other disturbances. GARA [13] combines these two paradigms to provide more graceful degradation and higher resource utilization at the system and application level. GARA uses priority based flow reservation which can be altered at run-time by both the application and by third parties on behalf of the application. This type of reservation scheme allows applications to monitor and react to changes in network capacity, while still attempting to ensure that high-priority flows can traverse the network. Furthermore, this type of reservation scheme is more amenable to dynamic flows which may only be active during a portion of time that the system is active. Statically defined slots reserved at design-time cause wasted resources by these applications whose flow is reserved but not used the entire time.

Finally, there do exist different protocols and communications paradigms which support run-time control of application network traffic, such as the Quality of Service (QoS) control mechanisms present in many implementations of OMG's Data Distribution Service (DDS) standard[27][26]. However, often the mechanisms available for controlling the QoS parameters of a given data stream are complex, interacting mechanisms which may be difficult for the application developer to understand and therefore are also not amenable to modeling and analysis at design time. Furthermore, the developers may not be provided with or have control over lower level implementation details such as the selected transport layer protocol, which may affect the available QoS or may not be fully supported by the infrastructure. Additionally, many of the available interaction paradigms either do not support design time QoS analysis with run-time monitoring and control or the supported QoS analysis and control interfaces are only informally specified.

CHAPTER III

DESIGN-TIME NETWORK PERFORMANCE ANALYSIS OF DISTRIBUTED CPS APPLICATIONS

Many CPS applications require networking of some form in order for the system to function nominally. This networking often performs a key role in the system, such as facilitating the communication and control of distributed sensors and control systems. Traditionally, these networks of CPS have been both isolated from external influences and predefined at system design-time. This isolation and pre-determination creates a static network with respect to both the topology of the network and the capacity of each network link. More recently however, CPS have become less isolated and more dynamic by utilizing heterogeneous and wireless networks and incorporating mobility.

Analyzing application and system network Quality of Service (QoS) requires either design-time models and analysis techniques or experimental measurements from an application and system testbed. For high- or mixed-criticality software and systems, typically experimental measurements are used but often these can be incomplete or quite costly to generate. Instead, a design-time modeling paradigm for networked applications and systems can provide developers and system-integrators the information to accurately predict the system and application network QoS.

3.1 Point-to-Point FDMA Network Analysis : COMPLETED

Some wireless mobile CPS networks, such as the network between a cluster of satellites orbiting Earth, vary periodically with respect to time, *e.g.* according to the cluster's orbital period. For such networks, the physical dynamics of the nodes in the cluster are well understood and predictable, therefore the network dynamics can be fairly predictable as well. For such predictable or periodic dynamic networks, the use of worst-case network performance for analysis and constraint verification wastes the network resources over much of the lifecycle of the system. Integrating the physical dynamics of the network into the modeling and analysis tools improves the performance of the systems without degrading its reliability.

FDMA, on the other hand, allocates frequencies in a preassigned communications frequency band to the nodes in the network. Since each node transmits on a different frequency, many nodes can transmit simultaneously, thus increasing their bandwidth in comparison to TDMA, but interference between transmissions on the multiple frequencies can cause packet collision and loss.

3.1.1 Problem

Current design tools do not incorporate the physical dynamics of the network for analysis of network constraints on the applications. Because of the diversity of CPS, IoT systems, and other networked embedded systems in general, modeling and analysis tools targeted towards these systems must support a wide range of configurations, architectures, standards, and interfaces. The same compatibility is required in network modeling and analysis frameworks. Because many of these systems may support different types of network communications hardware, often using multiple types of network interface hardware within the same system, the models of the network must be able to express network resources in a way that can capture these differences. One such standard for sharing a communications medium between multiple nodes is Frequency Division Multiple Access (FDMA). FDMA

is useful in wireless networks for providing simultaneous, collision-free communications capabilities to the nodes in the system.

3.1.2 Contributions

- We developed an analysis technique, with associated system and application network resource models for FDMA based networks, reported in [11].
- We demonstrated how to compose the models together and derive application and system network performance metrics at design-time, reported in [11].

3.2 Point-to-Point TDMA Network Analysis : COMPLETED

Medium channel access protocols are used in networking systems to govern the communication between computing nodes which share a network communications medium. They are designed to allow reliable communication between the nodes, while maintaining certain goals, such as minimizing network collisions, maximizing bandwidth, or maximizing the number of nodes the network can handle. Such protocols include Time Division Multiple Access (TDMA), which tries to minimize the number of packet collisions; Frequency Division Multiple Access (FDMA), which tries to maximize the bandwidth available to each transmitter; and Code Division Multiple Access (CDMA) which tries to maximize the number of nodes that the network can handle.

In TDMA, each node on the network is assigned one or more time-slots per communications period in which only that node is allowed to transmit. By governing these timeslots and having each node agree upon the slot allocation and communications period, the protocol ensures that at a given time, only a single node will be transmitting data, minimizing the number of collisions due to multiple simultaneous transmitters.

3.2.1 Problem

TDMA minimizes collisions and has an impact on the timing characteristics of the applications' network communications. Because of the diversity of CPS, IoT systems, and other networked embedded systems in general, modeling and analysis tools targeted towards these systems must support a wide range of configurations, architectures, standards, and interfaces. The same compatibility is required in network modeling and analysis frameworks. Because many of these systems may support different types of network communications hardware, often using multiple types of network interface hardware within the same system, the models of the network must be able to express network resources in a way that can capture these differences. One such standard for sharing a communications channel between multiple nodes is Time Division Multiple Access (TDMA). TDMA is useful in

both wireless and wired networks for providing guaranteed communications ability to all nodes in the system while addressing collision avoidance and bandwidth allotment in a deterministic manner.

3.2.2 Contributions

- We extended our system and application network resource models to encompass TDMA based networks, reported in [**?**].
- We derived formulas which abstracted away the TDMA scheduling to remove the requirement to explicitly incorporate the TDMA scheduling into the system and application models, reported in [**?**].

3.3 Network Performance Prediction Comparison with Other Analysis Tools

3.3.1 Problem

When developing a new analysis technique to predict application network performance, alternative techniques must be evaluated to determine the utility of the new techniques. Application developers and system integrators can then use these comparisons as a metric for choosing between the available analysis tools. For the tools and techniques to affect a meaningful change in system and application development, they must be shown to be more effective by some metric for at least certain classes of systems or applications.

3.3.2 Proposed Contributions

- We will develop test system and application models using our analysis techniques for which we can determine the predicted network resource requirements.
- We will develop those same test system and application models using other network performance analysis techniques, e.g. Network Calculus, for which we can determine comparison predicted network resource requirements.
- We will compare the predicted results for the test system and application combinations to see what, if any, difference exists between the techniques.

3.4 Application Network Profile Generation from Business Logic Models

3.4.1 Problem

Developing an accurate, precise network resource requirement profile for an application as a function of time is a challenging and daunting task for all but the simplest of applications. Analysis of the code to determine worst-case data production over certain intervals can provide a better approximation of the network resource requirements than traditional worst-case analysis over the entire lifetime of the application. However, this technique is not feasible for large or complex applications and may yield only a marginal increase in model fidelity. Another possible technique for creating the application's network resource requirement profile as a function of time is to run the application on the target system with integrated measurement facilities to more precisely and accurately determine the resource requirements. This technique however is impractical or infeasible for application developers who do not have direct access to the system, or for systems whose architectures or environments cannot be easily simulated.

3.4.2 Proposed Contributions

- We will develop an add-on to our currently existing modeling language for application business logic which captures the network resources required during each part of the business logic model.
- We will develop a compositional technique for generating the network resource requirement profile for an application from the combined business logic models of that application's components.
- We will develop test applications for our RCPS testbed which adhere to the business logic models and allow us to measure the accuracy and precision of the predictions using these generated profiles.

3.5 Statically Routed Network Analysis

3.5.1 Problem

As CPS become more distributed in nature and begin to act as infrastructure for distributed applications towards IoT systems, they will necessarily need to handle more network resource management and network connection routing within their network as well as between their own network and any external networks to which they are connected. Such networks generally rely on routing to allow more flexibility in the system with respect to node placement and connectivity. Adding routing to the network also has the effect of increasing the complexity of the network performance analysis and can cause drastic differences in application network performance when compared with networks without routing. Therefore the design-time analysis tools which help predict application network performance must take this routing into account.

3.5.2 Proposed Contributions

- We will extend our network resource modeling and analysis techniques to support networks in which system nodes can act as routers for packets in the network.
- We will show experimentally the validity of the analysis results using our RCPS testbed and test applications.

3.6 Mathematical Equivalence Analysis Comparing the Proposed Analysis with Network Calculus

3.6.1 Problem

Network Calculus has been around for a couple decades so far, and has seen numerous improvements and extensions. Mathematically analyzing the equivalence between the methods of the proposed analysis with those of Network Calculus would pave the way for enhancing the proposed work with similar extensions as have been developed for Network Calculus. This equivalence analysis should hold, since both techniques are based on the min-plus calculus dioid. By showing the equivalence, we would show that the proposed analysis techniques are still linear in nature and have similar system-theory based applications.

3.6.2 Proposed Contributions

- We will formally analyze the linearity of proposed analysis techniques
- We will mathematically analyze the differences between the proposed technique and Network Calculus, showing their equivalence with respect to functional composition

CHAPTER IV

RUN-TIME NETWORK PERFORMANCE MONITORING AND MANAGEMENT FOR DISTRIBUTED CPS APPLICATIONS

4.1 Experimental Verification of Performance Prediction : COMPLETED

4.1.1 Problem

When developing a new analysis technique to predict application network performance, verification that the results of the analysis are accurate is paramount. Experimental results are required not only to judge whether or not the theory is sound, but also to allow application developers and system integrators to judge the applicability of the analysis to their systems and applications. If the performance metrics have too low confidence, i.e. the error or variance in the predictions is too high, the analysis results may cease to be useful.

4.1.2 Contributions

- We have developed a network emulation testbed which applies the system's time-varying network profile to all network traffic in the system, reported in [11].
- We have developed application network traffic generation code which produces network traffic according to the supplied profile
- We have instrumented the senders and receivers to measure the delay, throughput, and buffer requirements
- We have collected data from applications and measured the differences between the measured QoS metrics and the predicted QoS metrics, reported in [11].

4.2 Network Resource Monitoring and Management Integrated into DREMS : COMPLETED

4.2.1 Problem

Distributed, deployed CPS which require design-time assurances of system stability and security with respect to resources and communications must have some protection against propagating software faults or malicious actors. One such avenue for fault or attack propagation is the system's communications network. Typically, systems act as the infrastructure for the applications, meaning they provide the computational resources, hardware access, and software communications and configuration required for the applications to function. In such architectural designs, the systems have little to no knowledge about the correct behavior of the application code[], and therefore must set resource and performance limits on applications based on a broader scope. Furthermore, these limits generally do not change with respect to time[], and must be even broader to account for whatever the applications may need to do over their lifetime. In general, resource requirement models for applications which are either not application specific or do not incorporate temporal behavior prove to be inefficient for specifying system resource limits which can provide system stability and ensure resource availability. A typical example of such types of resource reservation schemes backfiring is in Distributed Denial of Service (DDoS) attacks[], in which many compromised applications generate slightly more network traffic than usual (but still within their allowance) to generate a combined network traffic profile that can effectively take their target off of the network.

4.2.2 Contributions

- We integrated our network resource modeling techniques into a system and application analysis and development toolsuite. The tool verified that the system could provide all the network resources (which varied as a function of time) required by the applications.

- Into the application's generated middleware interface code we integrated measurement, detection, and mitigation code which (1) measured the characteristics of the application's network traffic, (2) detected if the application's network traffic exceeded its profile that was provided during system analysis, and (3) blocked all traffic from leaving application-space (i.e. it did not get into kernel-space) which was detected as having exceeded the profile.

- We showed that applications which tried to send more data onto the network than they had (at design-time) specified, were not allowed to send the spurious data.

4.3 Network Application Fault/Anomaly Classification

4.3.1 Problem

For distributed systems which must ensure resource availability and system stability, a key aspect of the infrastructure is detection and mitigation of faults or anomalies. With respect to network resources, examples include ensuring the validity of authorization, checking source and destination for communications to ensure authorized communication flows, etc. However, software glitches or compromised applications can abuse system resources that they have been allowed. As described in the previous section, higher-fidelity resource modeling and monitoring is required to prevent such faults or compromises from propagating throughout the system. However, mitigating the propagation only solves part of the problem; ideally the system should try to classify the type of fault or anomaly and begin diagnostics to trace the fault/anomaly back to its origin.

4.3.2 Proposed Contributions

- We will use our resilient CPS testbed to run distributed network tests to classify certain types of anomalies, e.g. DDoS attacks from compromised applications within the cluster.
- We will use the network resource utilization measurements gained from the tests to derive metrics which allow us to differentiate between classes of behavior, e.g. standard/stable application behavior vs. DDoS behavior.
- We will then use the classifications to show that the system can detect these types of attacks, mitigate their propagation, and report the attack to the system's manager.

CHAPTER V

CONCLUSIONS

We have described in this proposal the what aspects of Cyber-Physical Systems(CPS) analysis, design, development, and integration we are addressing. We have provided descriptions of the relevant related work in this area, covering both the design-time modeling, analysis, and performance prediction for networked, distributed, CPS applications and the run-time monitoring and management of application and CPS network resources. The subsequent chapters described the work that has been done thus far regarding both design-time modeling and analysis and run-time monitoring and management. The remaining challenges within the scope of this research have been explained in their respective chapters, describing the remaining work to extend the modeling and analysis techniques for more precise resource modeling and more relevant system modeling, and describing the remaining work to extend the run-time management to categorize and classify detected anomalies.

Table 1 provides the tentative timetable for the proposed research.

Table 1: Proposed Research Timetable

Comparison with other Analysis Tools	05/2015 - 06/2015
Network Profile Generation from Business Logic Models	06/2015 - 07/2015
Support for Statically Routed Networks	06/2015 - 07/2015
Proving Equivalence to Network Calculus Mathematically	05/2015 - 08/2015
Fault/Anomaly Classification	06/2015 - 08/2015

CHAPTER VI

PUBLICATIONS

6.1 Highly Selective Conference Papers

- **W. Emfinger**, G. Karsai, "Modeling Network Medium Access Protocols for Network Quality of Service Analysis". In proceedings of the 18th IEEE Symposium on Real-Time Computing (ISORC), 2015, Auckland, New Zealand (Acceptance Rate %).

6.2 Other Conference and Workshop Papers

- **W. Emfinger**, G. Karsai, A. Dubey, A. Gokhale, "Analysis, Verification, and Management Toolsuite for Cyber-Physical Applications on Time-Varying Networks (Work-in-Progress)". In proceedings of the fourth Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems (CyPhy), 2014, Berlin, Germany.

- **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.

- Pradhan S., **W. Emfinger**, A. Dubey, W. Otte, A. Coglio, D. Balasubramanian, A. Gokhale, G. Karsai, "Establishing secure interactions across distributed applications in satellite clusters". IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT), 2014, Laurel, MD, USA.

- Balasubramanian, D., **W. Emfinger**, P. S. Kumar, W. Otte, A. Dubey, and G. Karsai, "An application development and deployment platform for satellite clusters", 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", IEEE International Symposium on Rapid System Prototyping (RSP): IEEE, 10/2014.

- Dubey, A., **W. Emfinger**, A. Gokhale, G. Karsai, W. R. Otte, J. Parsons, C. Szabo, A. Coglio, E. Smith, and P. Bose, "A Software Platform for Fractionated Spacecraft", 2012 IEEE Aerospace Conference, Big Sky, Montana, 03/2012

- 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", IEEE Software, vol. 99: IEEE Computer Society, 2014.

6.3 Submitted Papers - Awaiting Reviews

- **W. Emfinger**, P. Kumar, A. Dubey, G. Karsai, "Towards Assurances in Self-Adaptive, Dynamic, Distributed Real-time Embedded Systems". Software Engineering for Self-Adaptive Systems: Assurances, 2015.

REFERENCES

[1] *Stochastic Network Calculus*. Springer London, London, 2009.

[2] S. Blake, S. Microsystems, and Z. Wang. An Architecture for Differentiated Services. *RFC*, pages 1–37, 1998.

[3] A. Bouillard, N. Farhi, and B. Gaujal. Packetization and packet curves in network calculus. In *Proc. 6th Int Performance Evaluation Methodologies and Tools (VALUETOOLS) Conf*, pages 136–137, 2012.

[4] R. Braden, D. Clark, and S. . Shenker. Integrated Services in the Internet Architecture : an Overview. *IETF RFC 1633, July*, pages 1–28, 1994.

[5] A. Burchard, J. Liebeherr, and S. Patek. A min-plus calculus for end-to-end statistical service guarantees. *IEEE TRANSACTION ON INFORMATION THEORY*, 52:4105–4114, 2006.

[6] M. Carbone and L. Rizzo. Dummynet revisited, 2010.

[7] O. S. Consortium et al. Openflow switch specification version 1.0. 0, 2009.

[8] R. Cruz. A calculus for network delay. II. Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, 1991.

[9] R. L. Cruz. A calculus for network delay–I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, 1991.

[10] S. Dobson. An adaptive systems perspective on network calculus, with applications to autonomic control. *International Journal of Autonomous and Adaptive Communications Systems*, 1(3):332, 2008.

[11] W. Emfinger, G. Karsai, A. Dubey, and A. Gokhale. Analysis, verification, and management toolsuite for cyber-physical applications on time-varying networks. In *Proceedings of the 4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*, CyPhy '14, pages 44–47, New York, NY, USA, 2014. ACM.

[12] T. Ferrandiz, F. Frances, and C. Fraboul. A Network Calculus model for SpaceWire networks. In *Proceedings - 17th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 2011*, volume 1, pages 295–299, 2011.

[13] I. Foster, A. Roy, and V. Sander. A quality of service architecture that combines resource reservation and application adaptation. *Quality of Service, 2000. IWQOS. . . .*, 2000.

[14] G. R. Garay, J. Ortega, and V. Alarcon-Aquino. Comparing Real-Time Calculus with the existing analytical approaches for the performance evaluation of network interfaces. In *CONIELECOMP 2011 - 21st International Conference on Electronics Communications and Computers, Proceedings*, pages 119–124, 2011.

[15] G. Giambene. *Queuing Theory and Telecommunications Networks and Applications*. 2005.

[16] Gilberto Flores Lucio, Marcos Paredes-farrera, Emmanuel Jammeh, Martin Fleury, and M. J. Reed. OPNET Modeler and Ns-2: Comparing the Accuracy of Network Simulators for Packet-Level Analysis using a Network Testbed. *3rd WEAS International Conference on Simulation, Modelling and Optimization (ICOSMO)*, pages 700–707, 2003.

[17] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K.-c. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. Effects of Detail in Wireless Network Simulation. In *Proceedings of the{SCS} Conference on Communication Networks and Distributed Systems Modelling and Simulation*, pages 3–11, 2001.

[18] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the air: Practical wireless network coding. *IEEE/ACM Transactions on Networking*, 16(3):497–510, 2008.

[19] D. G. Kendall. Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain. *The Annals of Mathematical Statistics*, 24(3):338–354, 1953.

[20] K. Khan and H. Peyravi. Delay and queue size analysis of tdma with general traffic. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1998. Proceedings. Sixth International Symposium on*, pages 217–225, Jul 1998.

[21] J.-Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag, Berlin, Heidelberg, 2001.

[22] H. Li, X. Liu, W. He, J. Li, and W. Dou. End-to-End Delay Analysis in Wireless Network Coding: A Network Calculus-Based Approach. *2011 31st International Conference on Distributed Computing Systems*, pages 47–56, 2011.

[23] J. D. C. Little. A proof for the queuing formula: L = Îżw. *Operations Research*, 9(3):383–387, 1961.

[24] I. Mahadevan and K. Sivalingam. Quality of Service architectures for wireless networks: IntServ and DiffServ models. *Proceedings Fourth International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99)*, 1999.

[25] D. Mahrenholz and S. Ivanov. Real-time network emulation with ns-2. *Proceedings*

- *Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications, DS-RT 2004*, pages 29–36, 2004.

[26] Object Computing Incorporated. OpenDDS. http://www.opendds.org, 2007.

[27] Object Management Group. *Data Distribution Service for Real-time Systems Specification*, 1.2 edition, Jan. 2007.

[28] L. Rizzo. Dummynet : a simple approach to the evaluation of network protocols. *ACM SIGCOMM Computer Communication Review*, 27(1):31–41, 1997.

[29] J. B. Schmitt, F. A. Zdarsky, and L. Thiele. A comprehensive worst-case calculus for wireless sensor networks with in-network processing. In *Proceedings - Real-Time Systems Symposium*, pages 193–202, 2007.

[30] T. J. Schriber, D. T. Brunner, and J. S. Smith. Inside discrete-event simulation software: How it works and why it matters. In *Proceedings of the 2013 Winter Simulation Conference - Simulation: Making Decisions in a Complex World, WSC 2013*, pages 424–438, 2013.

[31] L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *in ISCAS*, pages 101–104, 2000.

[32] A. Varga and R. Hornig. An overview of the omnet++ simulation environment. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[33] Y. Wang and T. Wang. Applying stochastic network calculus to 802.11 backlog and delay analysis. In *IEEE International Workshop on Quality of Service, IWQoS*, 2011.

[34] H. Xiao and W. Seah. A flexible quality of service model for mobile ad-hoc networks. . . . *Proceedings, 2000. VTC* . . . , 2000.