Analysis, Verification, and Management Toolsuite for Cyber-Physical Applications on Time-Varying Networks (Work in Progress)

William Emfinger,Gabor Karsai,Abhishek Dubey,Aniruddha Gokhale
Institute for Software-Integrated Systems
Vanderbilt University
Nashville, TN 37235
{emfinger,gabor,dabhishe,gokhale}@isis.vanderbilt.edu

ABSTRACT

Cyber-Physical Systems (CPS) are increasingly utilizing advances in wireless mesh networking among computing nodes to facilitate communication and control for distributed applications. Factors such as interference or node mobility cause such wireless networks to experience changes in both topology and link capacities. These dynamic networks pose a reliability concern for high-criticality or mixed-criticality systems which require strict guarantees about system performance and robustness prior to deployment. To address the design- and run-time verification and reliability concerns created by these dynamic networks, we are developing an integrated modeling, analysis, and run-time toolsuite which provides (1) network profiles that model the dynamics of system network resources and application network requirements over time, (2) design-time verification of application performance on dynamic networks, and (3) management of the CPS network resources during run-time. In this paper we present the foundations for the analysis of dynamic networks and show experimental validations of this analysis. We conclude with a focus on future work and applications to the field.

Keywords

Verification, QoS, CPS, cyber-physical, networks, managed distributed systems.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification—*Reliability*

General Terms

Verification[1]

1. INTRODUCTION

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.

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.

However, our current design tools do not incorporate the physical dynamics of the network for analysis of network constraints on the applications. Towards these goals, we have developed time-varying network traffic models that can be specified by the application developer and analyzed against the system network profiles, specified as $[time, data]$ series. By analyzing these profiles, we can determine exactly how the system will transfer the application data. Analysis of this transmitted data profile vs. the application data profile provides the developer with information about both the minimum buffer required for application communication without loss of information and the maximum buffer delay in communication caused by the network. Metrics such as these allow the developer to guarantee in design time that the application will not exceed its memory and latency constraints at run-time.

2. RELATED WORK

Since computing networks are so prevalent, many tools exist to analyze system network behavior, either through simulation or abstract modeling. OMNET++[1] accurately 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. Because it focuses on actually simulating the network, OMNET++ is less useful for providing design-time guarantees about the performance, and it becomes impractical to use for large, complex, cyber-physical systems that host many applications distributed throughout the nodes.

Network Calculus[2], on the other hand, focuses on abstracting the network traffic and the computing nodes as arrival curves and traffic shapers. The arrival curves and service curves model the amount of data generated or serviced in a given window of time and are bounded by maximum and minimum arrival and service curves. 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. These deterministic constraints can be relaxed so that the arrival and service curves can be probabilistic, causing the bounds on the performance to be probabilistic as well[3]. 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.

Network Calculus' principles and the underlying mathematics of max-plus calculus were adapted in [4] into Real-Time Calculus for modeling and analyzing computational requirements and resources in real-time systems. Using the concept of capacity request and delivery curves adapted from Network Calculus, bounds on the computation time and scheduling could be derived along with the ability to test schedulability of the applications on the real-time system.

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[5][6]. 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. Additionally, many of the available interaction paradigms either do not support design time QoS analysis and run-time monitoring and control or the supported QoS analysis and control interfaces are only informally specified.

3. CONTRIBUTIONS

To model the network capability of the system and the application traffic patterns, we have developed a network modeling paradigm similar to Network Calculus' traffic arrival curves and traffic shaper service curves.

Similarly to Network Calculus' arrival curves and service curves, our network profiles model how the network performance or application traffic generation changes with respect to time. Whereas Network Calculus' modeling transforms application data profiles and network service profiles into min and max curves for data received vs. size of time-window, our models take a simpler, deterministic approach which models exactly the data generated by the application and the data which could be sent through the network, allowing our performance metrics to be more precise. Specifically, the bandwidth that the network provides on a given communication link is specified as a time series of scalar bandwidth values. Here, bandwidth is defined as data rate, i.e. bits per second, over some averaging interval. This bandwidth profile can then be time-integrated to determine the maximum amount of data throughput the network link could provide over a given time. The bandwidth profile for the application traffic similarly can be time-integrated to determine the amount of data that the application attempts to send on the network link as a function of time.

Having time-integrated the bandwidth profiles to obtain data vs. time profiles that the application requires and that the system provides, we can use convolution ($\otimes$) on these two profiles to obtain the transmitted link data profile as a function of discrete time. The convolution we define on these profiles borrows concepts from the min-plus calculus used in Network Calculus, but does not use a sliding-window and instead takes the transformed minimum of the profiles. For a given application data generation profile, $r[t]$, and a given system link capacity profile $p[t]$, where $t \in \mathbb{N}$, the link transmitted data profile $l[t]$ is given by the convolution Equation 1. The difference $(p[t-1]-l[t-1])$ represents the difference between the amount of data that has been transmitted on the link ($l[t-1]$) and the data that the link could have transmitted at full utilization ($p[t-1]$). As demonstrated by the convolution equation, $\forall t : l[t] \leq r[t]$, which is the relation that, without lower-layer reliable transport, the link cannot transmit more application data for the application than the application requests as there will be packetization and communication header overhead as well.

$$\begin{aligned} y = l[t] &= (r \otimes p)[t] \\ &= min(r[t], p[t] - (p[t-1] - l[t-1])) \end{aligned} \tag{1}$$

$$\text{buffer} = sup\{r[t] - l[t] : t \in \mathbb{N}\} \tag{2}$$

$$\text{delay} = sup\{l^{-1}[y] - r^{-1}[y] : y \in \mathbb{N}\} \tag{3}$$

Equation 2 and Equation 3 calculate, using $l[t]$, $r[t]$, and the inverse function $l^{-1}[y]$, the minimum buffer size required for the application and the maximum buffering delay experienced by application data, respectively. Figure 1 depicts the convolution operation and shows a schematic representation of the maximum buffer delay and the minimum buffer size. As can be seen in the figure, the maximum horizontal distance between the required profile and the link profile is equal to the maximum buffer delay, while the maximum vertical distance is equal to the minimum buffer size required for loss-less transmission of data on the link.

To verify the validity of these operations and metrics, we

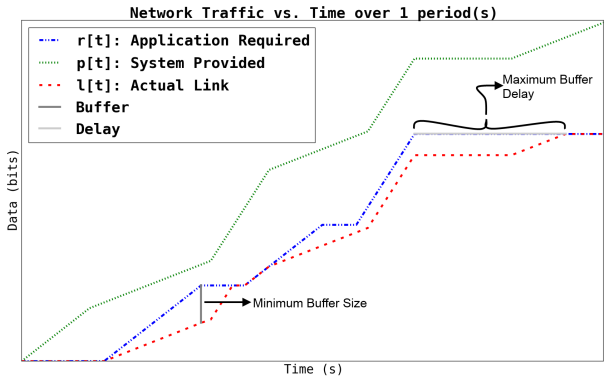

Figure 1: Representative example demonstrating convolution of the data vs. time profiles that the application requires and that the system provides. The resultant data vs. time profile describes the data that the system actually transmitted on the link.

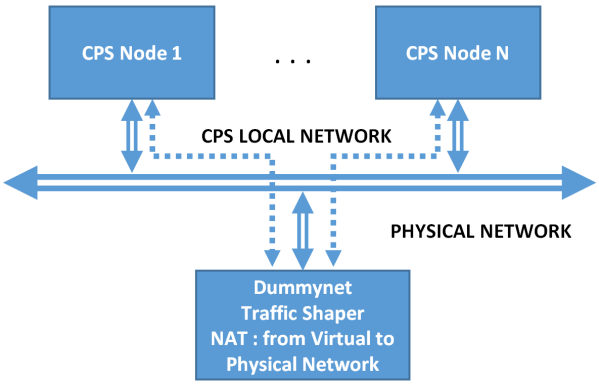

Figure 2: Diagram of the testbed network and setup. The CPS nodes are connected to each other, but their network communication is routed through the traffic shaping and delay node.

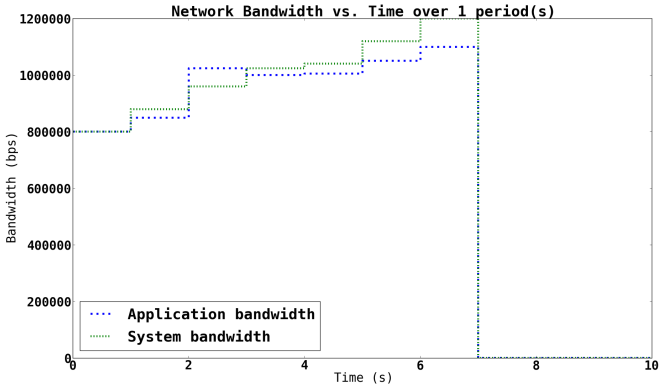

Figure 3: Bandwidth profiles for both what the application requires and what the system provides as a function of time.

developed network measurement code which generated data for the network according to the supplied profile. This code executed on a private network testbed of nodes connected to each other through a gigabit Ethernet switch. We implemented the network link profile using a traffic shaping

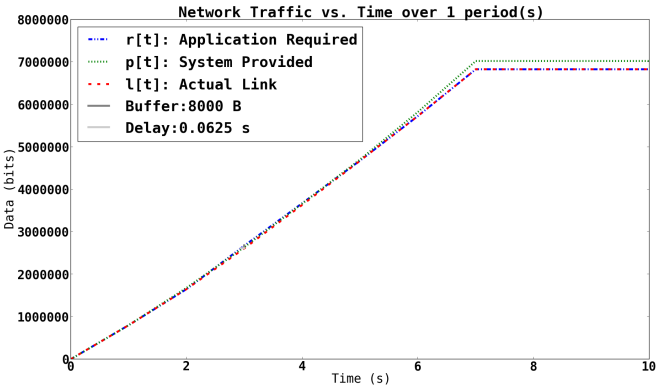

Figure 4: These data vs time profiles are the time-integration of the bandwidth vs time profiles from Figure 3. The resultant Link profile represents the convolution of the two data profiles. N.B. this plot shows the full data vs time profiles; Figure 5 magnifies this plot for visibility of the buffer and delay.

node on this network through which all the application traffic flowed. On this node we ran dummynet[7], which can be configured to control bandwidth, latency, and packet loss on a per-link basis. For these experiments, we configured the traffic shaping node to control the bandwidth of the specified link according to the system provided network profile. This testbed setup is shown in Figure 2.

Using this testbed system and performance code, we were able to measure the effective bandwidth over time to ensure that, at all times, the resultant traffic profile matched the predicted traffic profile. In addition, the sending and receiving time for each message was recorded to allow calculation of the buffer delay. Finally, by analyzing the measured data profiles in the same way that we analyzed the predicted data profiles, we can calculate the minimum buffer size required for the application on the network.

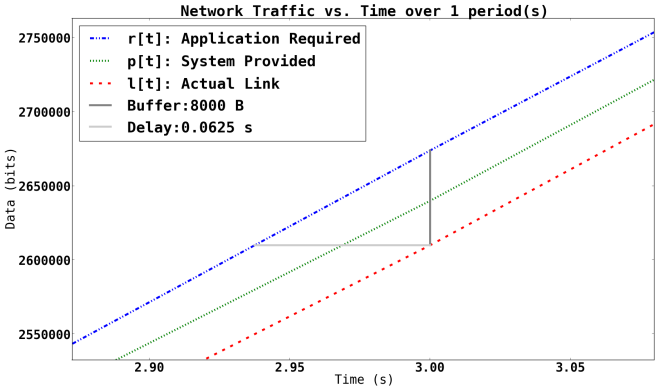

Figure 5: The maximum buffer delay and minimum network buffer required are shown zoomed-in for clarity from Figure 4. The vertical line shows the minimum buffer required, while the horizontal line shows the maximum buffer delay incurred.

Time-integrating the bandwidth profile in Figure 3, we obtain the data profiles (Figure 4). These data profiles are convolved as in Equation 1 to produce the link data profile.

From the link data profile and the application required profile we can predict the delay and buffer bounds, which are zoomed-in for display in Figure 5. For the sample profile, the predicted versus measured maximum buffer delay, time of delay, and minimum buffer size are shown in Table 1. As the table shows, the measurements from the experiments closely correlate with the model's predictions, with the predictions erring on the conservative side.

Table 1: Network utilization calculations and measured results using UDP over IPv6.

	Predicted	Measured (μ, σ)
Buffer Delay (s)	0.0625	(0.06003 , 0.00029)
Time of Delay (s)	3.0	(2.90547 , 0.00025)
Buffer Size (bytes)	8000	(7722.59 , 36.94)

4. FUTURE WORK

This work's goal is to provide a design-time analysis of dynamic networks for verification of application performance. Towards that goal, we have shown an analyzable network model for deterministic time-varying networks. We have shown the validity of this model and its analysis through experimental results on a time-varying network. More work is needed however to extend this model for deployment in a run-time system.

To extend this analysis into the run-time system we are modifying the component connector layer of the middleware to measure the run-time performance of the application. In this layer, the application's network traffic can be monitored and controlled. The monitoring of the network traffic in this layer allows us to ensure that each application gets the network traffic profile it was guaranteed while also providing the developer with a common interface for controlling the QoS on the network regardless of the interaction pattern being used. Conversely, the control of the network traffic further allows us to ensure that any application's attempts to exceed its traffic profile limits do not affect another application's utilization of the network.

A few of the key features on which we are currently working include

- Extending the network profile modeling towards dynamic networks whose evolution may not be completely known at design time
- Determining the scalability of the analysis for larger networks
- Performing constraints analysis between the applications and the network links to determine if applications can execute on the platform successfully
- Integrating schedulability analysis to provide information to inadmissible applications about how they need to reschedule their network traffic for admission to the network
- Business logic modeling to generate the application traffic profiles
- Run-time network traffic management code integrated into the middleware for providing the network queues, monitoring bandwidth and delay, and ensuring conformance of applications to their traffic profiles
- Extending the run-time management to include dynamic network load balancing and traffic profile rescheduling based on availability of the network

5. CONCLUSIONS

In our preliminary work, we have integrated dynamic network analysis into CPS application design, analysis, and development tools. For systems which communicate over a dynamic network, such as a cluster of satellites orbiting a celestial body, the network analysis and prediction at design-time typically focuses on worst-case scenarios and worst-case performance. This focus on worst-case performance of the network is a useful method for providing guarantees about application and network performance, but it may waste available network resources during periods of good connectivity. To minimize the wasted system resources involved with the varying network performance, we have begun developing a network analysis model which takes into account the time-variance of the network with the goal of predicting how both the application and the system perform over time. We have shown that in simple deterministic cases, the network model correctly predicts useful metrics like delay and buffer bounds using application test code executing on a cluster testbed. Our goals are now to extend the model to allow for better guarantees in more variable networks, to accurately generate the system and application profiles at design time for analysis, and to provide integrated Quality-of-Service control within the application communications layer of the run-time infrastructure. Using these tools, we hope that cyber-physical systems application developers and system integrators will be able to not only have greater confidence in system functionality and stability, but also gain more run-time performance from the system and its applications.

Acknowledgments: The DARPA System F6 Program and the National Science Foundation (CNS-1035655) 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 or NSF.

6. REFERENCES

[1] 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*, (ICST, Brussels, Belgium, Belgium), pp. 1–10, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

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

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

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

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

[6] Object Computing Incorporated, "OpenDDS." `http://www.opendds.org`, 2007.

[7] M. Carbone and L. Rizzo, "Dummynet revisited," *SIGCOMM Computer Communication Review*, vol. 40, pp. 12–20, Apr. 2010.