Analyzing the Performance of the Network Protocols Based on System-Level Modeling*

Linbo Fang\textsuperscript{1,2}, Zhangqin Huang\textsuperscript{1}, and Yibin Hou\textsuperscript{1}

\textsuperscript{1}School of Computer Science, Beijing University of Technology, Beijing 100022, P.R. China
\textsuperscript{2}The Fourth department, the Second Artillery Engineering College, Xi’an 710025, P.R. China

Summary

System-level design methods enable developing an executable model which allows the qualitative and quantitative properties to be analyzed before the system is actually realized. The modeling language Parallel Object-Oriented Specification Language (POOSL) has been proven to be very useful for system-level modeling when we want to describe concurrent and distributed behaviors effectively. The network protocols have many complex concurrent and distributed characteristics, thus it is very difficult to be analyzed by mathematical methods. One of the most promising solutions to this problem is system-level modeling and simulation. A number of parameters in the network protocol have a major effect on the overall performance. To be able to evaluate the effect, a system-level model of the network protocol is built in POOSL. Through the simulation of the model, results of the performance analysis on the influence of the parameters can be obtained. Furthermore, some key-parameters can be determined during the earliest phases of the design. This paper describes the experiences of using system-level modeling to analyze the performance properties of Sliding Window Protocol (SWP). The system-level model of SWP is built and the performance properties are investigated for a number of configurations of the system parameters.

Key words:
System-Level Design, Parallel Object-Oriented Specification Language (POOSL), System-Level Modeling, Performance Analysis, Sliding Window Protocol

1. Introduction

Industry is facing the challenge of designing hardware/software systems of growing complexity, increasing implementation costs and time-to-market imperative [1]. To manage complexity and to shorten design cycles, industry is forced to apply system-level design methods to making well-founded decisions in the earliest phases of the design process. System-level design methods are frameworks for structuring the earliest phases of the design process with the intention to find a feasible design before actually realizing a system in terms of hardware and software components [2, 3, 4].

System-level design methods should be based on well-defined modeling language to support developing system-level model that adequately presents the system to be designed. POOSL (Parallel Object-Oriented Specification Language) [5, 6] is an effective modeling language which enables to construct succinct executable models of complex hardware/software system [7]. It has been proven to be very useful for modeling the real-life industrial-sized systems such as Packet Switch System [8], Network Processor [9] and DECT Wireless Protocol [10].

Network protocols which aim at reliable transmission of data over unreliable channels are deeply studied in computer science. They usually involve a subtle interaction of a number of distributed components and have a high degree of parallelism, so it’s very difficult to analyze their performance characteristics by mathematical analysis [11]. The most promising solutions to this problem are the use of system-level modeling and simulation.

One of the most efficient protocols for reliable transmission is the Sliding Window Protocol (SWP) [12]. Many popular network protocols such as TCP and HDLC are based on the SWP. A number of key-parameters like the window size, the timeout period and the packet size have a major effect on the overall performance in the SWP. How to determine the key-parameters before the implementation of the protocols is important while designing novel protocols with the sliding window algorithm. In this paper, the system-level model of SWP is created and the performance properties are investigated for a number of configurations of the system parameters.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of the POOSL language and tools. System-level design method is introduced in section 3. In section 4, the concepts of the selective-repeat sliding window protocol are formulated together with some performance issues. The system-level model of the selective-repeat sliding window protocol is created. The simulation results are illustrated in section 5. Finally, we present our conclusions and directions for future research.

* This research is supported by the National Natural Science Foundation of China under Grant No.90407017 and the Natural Science Foundation of Beijing of China under Grant No.KZ200510005006.
2. POOSL Language and Tools

2.1 The POOSL Language

POOSL is an expressive system-level modeling language based on a small set of powerful language primitives. Based on a mathematically defined semantics [5, 6], POOSL enables a precise representation of the system. It consists of a data part and a process part. The data part is based on traditional object-oriented programming languages like C++. The process part is based upon real-time probabilistic extension of the process algebra CCS [13]. When modeling a system by POOSL, one is working with three entities: process objects, data objects and clusters.

Data objects can describe complex functional behavior. They are contained in process objects and they model the private data of these process objects. Process objects represent the elementary time-related behavior of the system. Process objects contain internal data in the form of data objects that are stored in instance variables. They communicate by passing messages over static channels synchronously. Behavior of process objects is described by statements, such as procedure abstraction, parallel and sequential composition, repetition and (conditional or non-deterministic) selection statements, synchronous (conditional) message (and data object) passing primitives, data expressions, guarded commands, abort and interrupt primitives and the delay primitive.

Clusters consist of process objects and other clusters. They are used to create a hierarchical structure of process objects and other clusters and hide their internal structure.

2.2 The SHESim Tool and the Rotalumis Execution Engine

To support the creation of executable model, the SHESim tool [6, 14] was developed using C++. SHESim is an interactive modeling and simulation tool. It is used to incrementally specify and modify POOSL models. To inspect the history of messages that have been exchanged between different entities, interaction diagrams can be generated automatically during the simulation. Rotalumis is an execution engine that can run large models of industrial-sized systems at high speed. Generally, we execute the model with Rotalumis after the model has been validated in SHESim. The performance analysis results can be obtained during the simulation execution by Rotalumis.

3. System-Level Design Method

System-level design concerns developing the main concepts for a system and with evaluating whether these concepts will lead to the satisfaction of all functional and non-functional requirements. The ultimate aim of the system-level design is to tackle the difficulty of making the right decisions during the earliest design phases. System-level design flow is shown in fig. 1.

![Fig. 1 System-level design flow](image)

Starting from system need and some specification of the desired functionality, designers may come up with several concepts for realizing this functionality such that the specified requirements are expected to be satisfied. Based on the concepts and requirements, the conceptual model can be obtained through formulation.

To investigate whether the effects of the proposed concepts are indeed as expected, behavior executable models are developed. The construction of behavior model involves making abstractions from implementation details since these details cannot be known in advance. From the behavior model, we can analyze whether the system behaviors are correct. If the design satisfied all functional requirements, the behavior model can be extended with performance evaluation. The extended model allows evaluate whether the design solutions for realizing the functionality will satisfy the non-functional performance requirements. Some design key-parameters can be determined through simulation of the extended model. Also, the bottleneck of the system can be identified.

If all requirements of certain design alternatives are satisfied according to the simulation of the abstract model, these alternatives may serve as a basis of realizing the system. A specific alternative may be chosen if it satisfies the requirements in a better way compared to other alternatives.
In next section, the concepts of the selective repeat sliding window protocol are formulated together with some performance issues. Based on the conceptual model, the system-level model is created.

4. System-Level Modeling

In this section, we create the system-level model of the selective-repeat sliding window protocol (SR SWP). Before the POOSL model was attempted, the conceptual model was constructed so as to provide an overview of the protocol.

4.1 Conceptual Model

According to the error disposal method of SR SWP [12], we create its conceptual model.

![Fig. 2 Conceptual model of the SR SWP](image)

The conceptual model of the SR SWP is illustrated in fig. 2. It consists of (1) the network layer of the sender, (2) the data link layer of the sender, (3) the physical layer, (4) the data link layer of the receiver, (5) the network layer of the receiver.

In the beginning of communication, the sending window and the receiving window are empty. Whenever a new packet arrives at the sender from the network, the sender puts the packet and the sequence number with some control information into a frame, and then sends the frame to the frame channel. The control information includes frame type, frame sequence number and the acknowledgement number. After transmitting a frame, the sender stores the frame in the sending buffer of the data link and starts a new timer in the meantime. And then transmit the next data frame to the frame channel until the sending window is filled up. At the same time, the sender is waiting for the acknowledgement arriving from the network. If a valid acknowledgement comes in, the sender fetches the next packet from the network layer and put it in the buffer, overwriting the previous frame.

Every frame is associated with a timer. When the timer expires, it sends back a Negative Acknowledgment (NAK) frame to the sender. The sender will only retransmit the damaged frame until the frame is received by receiver successfully, and then the right sequential frames will send to the network layer. If the received frame is not the expected one, but its sequence number fall in the receiving window and it wasn't received before, the data link layer of the receiver will store the frame in the receiving buffer.

We investigate the following two performance issues of the SR SWP:

1) Maximizing the available bandwidth.
2) Minimizing the delay of the acknowledgements.

Some key-parameters in the SR SWP are as follows.

- **Packet size** ($S$): the size of information unit exchanged between the network layer and the data link layer in the sender or receiver.
- **Maximum window size** ($W$): the maximal number of the unacknowledged frames in the sending window or in the receiving window.
- **Timeout period** ($T_t$): the timeout period of a timer.
- **Bandwidth** ($B$): the maximum amount of data passing the channel at a given time. It depends on the medium capacity of the communication channels.
- **Process time of a packet** ($T_s$): the time that it takes for a packet to send/ receive to/from the channel, so $T_s = S/B$.
- **Channel delay and round trip time** ($T_r$): channel delay is the time it takes for data to travel across the channel. Round trip time is the time that it takes for data to travel across the channel from the sender to the receiver and then back. In this paper, we suppose that the channel delay of the data frames and acknowledgement frames have the same normal distribution with $\mu = T_r / 2$ for various variance ($\delta$). $\delta$ has different values such as $0.1* T_s$, $0.2* T_s$, or $0.5* T_s$.
- **Probability of Frame loss** ($F_l$): the probability of frame loss in the data frame channel or in the acknowledgement frame channel.

4.2 System-Level Model of SR SWP

Based on the conceptual model, we use SHESim tool to create the system-level model of fig. 3 using POOSL. The left is the side of the sender and the right is the receiver. Each side consists of three objects. They are network layer process, data link layer cluster and physical process. Network layer process is linked to data link layer cluster through a channel. The internal structure of the data link layer cluster is shown in fig. 4. It consists of a data link process and a timer process. The data link process communicates with the timer process by a channel.
In fig. 3, the data link layer cluster of the sender in the left fetches a packet from the network layer process via the channel ‘NLtoDLL’, put it to a frame with control information such as FrameKind, SeqNum and AckNum, and then sends it to the physical layer process via the channel ‘DLLtoPL’. In the mean time, the data link layer cluster of the receiver in the right is waiting for receiving frames from the physical layer process. After receiving a frame, the receiver will check whether the SeqNum of the frame falls in the receiving window. If the timer expires, the receiver sends back a NAK frame and the sender will retransmit the error frame when receiving the NAK frame. If the received frame is the expected one of the receiver, the right sequential frames will send to the network layer process.

A. Network Layer Process

In the model, the network layer process sends and counts the packets to the data link layer cluster via the channel ‘NLtoDLL’. In the meantime, it is waiting for receiving packets from the data link layer cluster through the port ‘conDLL’.

B. SR SWP process

The SR SWP process is the core in the model. The main functions of the SR SWP process are as follows.

1. Fetches packets from the network layer process via the channel ‘NLtoDLL’ and then forms frames with control information.
2. Manages the sending window and the receiving window according to the mechanism of the SR SWP.
3. Transmits and retransmits the data/acknowledgement frames using selective repeat to the physical layer process via the channel ‘DLLtoPL’.

C. Timer Process

After the sender transmits a frame, the timer process receives an event of starting a new timer from the SR SWP process, and then it starts the new timer. When a timer expires, the timer process in the model produces a timeout message to inform the SR SWP process via the channel ‘DLLtoTimer’.

D. Physical Layer Process

The physical layer Process is a bidirectional and unreliable channel. The packets in the channel will be delay a time. We suppose the delay time as a normal distribution with $\mu = \frac{T_r}{2}$ for various variance ($\delta$), $\delta$ is equal to $0.1*Ts$, $0.2*Ts$ and $0.5*Ts$ respectively.

4.3 Model Validation

The validation of POOSL models is supported by the interaction diagram tool [5] which allows designers to inspect every simulation step. Furthermore, SHESim offers the possibility to inspect the history of messages that have been exchanged between different entities in the form of a so-called interaction diagram. Interaction diagrams can be generated automatically during a simulation.

5. Performance Analysis

In this section, we illustrate simulation results derived from the simulation of the system-level model. Our experiments concentrate on the impacts of a number of parameters of the protocol on the available bandwidth. These parameters include the medium bandwidth, the window size, the timeout period, the packet size and the probability of frame loss. The configurations of the system parameters are given in table 1.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>System parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration 1</td>
<td>Bandwidth (10M): $T_r=100\text{ms}$, $T_t=120\text{ms}$, $S=1024\text{bytes}$;</td>
</tr>
<tr>
<td>Configuration 2</td>
<td>Bandwidth (100M): $T_r=10\text{ms}$, $T_t=12\text{ms}$, $S=1024\text{bytes}$;</td>
</tr>
<tr>
<td>Configuration 3</td>
<td>Bandwidth (1000M): $T_r=1\text{ms}$, $T_t=1.2\text{ms}$, $S=1024\text{bytes}$.</td>
</tr>
<tr>
<td>Configuration 4</td>
<td>$T_r=10\text{ms}$, $T_t=12\text{ms}$, $S=1024\text{bytes}$, $W=5-150$.</td>
</tr>
<tr>
<td>Configuration 5</td>
<td>$T_r=10\text{ms}$, $T_t=10-30\text{ms}$, $S=1024\text{bytes}$, $W=100$.</td>
</tr>
</tbody>
</table>
| Configuration 6 | $T_r=10\text{ms}$, $T_t=12\text{ms}$, $S=50-25000\text{bytes}$, $W=10$.

Configuration 1: the available bandwidth in different medium bandwidth without frame loss

To present the performance of the SR SWP, we ignore the possibility of the frame loss in the beginning. Fig. 5, Fig. 6 and Fig. 7 illustrate three simulation results in different medium bandwidth environments.
We can get some conclusions from the simulation:
(1) The window size is the crucial factor of the protocol performance without frame loss.
(2) In order to keep the pipeline busy, we can set the maximum window size as $\frac{T_s}{T_t}=\frac{T_s}{S/B}=122$. When the window size is around 125, the throughput of the SR SWP is growing to maximum.
(3) When the window size increases to $W=\frac{T_s}{T_t}=\frac{T_s}{S/B}=146$, the throughput will be dropped rapidly.
(4) The bandwidth utilization is 50%. Because the receiver uses the policy of the piggybacked acknowledgement, when a data frame arrives to the receiver, the acknowledgement cannot be generated immediately.

Configuration 2: the impacts of the different probabilities of frame loss on the available bandwidth

Fig. 8 shows the impacts of different probabilities of frame loss on the available bandwidth in a 100Mbps Ethernet LAN.

When the probability of the frame loss is 1%, the available bandwidth increases with the increasing of the window size and the optimal window size is around 125. When the probability of the frame loss is 2%, in the begging, the available bandwidth increases with the increase of the window size, but when the window size exceeds 75, the throughput is not growing, so the optimal window size is around 75. In the same way, when the probability of the frame loss is 5%, the optimal size is around 25. These results are valuable for designers to determine the receiving buffer size of data link layer when they design the network protocols on unreliable channels.

Configuration 3: the impacts of the different timeout periods on the available bandwidth

Timeout plays a crucial role in the SR SWP. Fig. 9 shows the impacts of timeout periods on the available bandwidth in a 100Mbps Ethernet LAN. For reliable communication channels, a high timeout period doesn’t reduce throughput. But for unreliable channels, a high timeout period will affect the performance unfavorably.

Configuration 4: the impacts of the different packet sizes on the available bandwidth

The impacts of different packet sizes on the available bandwidth in a 100Mbps Ethernet LAN are illustrated in Fig. 10.

A big enough packet generally gives better performance. However, if too big size packet fills up the pipeline, the performance will drop dramatically.

6. Conclusions and Future Research

To analyze the performance of the network protocols, the system-level modeling methods are introduced in this
work. System-level modeling methods define frameworks for developing an executable model of the system, which describes the incorporated functionality in the earliest phases of the design. The executable model allows evaluating whether the initial design decision will satisfy the requirements of the system or not before implementing the system with hardware and software components. Some key-parameters can be obtained through the simulation and performance analysis. Based on system-level modeling, the system-level model for the Selective-Repeat sliding window protocol is built. Results of the quantitative performance analysis on the influence of the parameters such as the window size, the packet size and the timeout period are given.

In our future work, more complex network environments like TCP/IP should be investigated. How to obtain the optimal parameters during the simulation is another direction.

Acknowledgment

This research is supported by the Research Plan of the National Natural Science Foundation of China under Grant No. 90407017 and the Natural Science Foundation of Beijing of China under Grant No. KZ200510005006.

References


Linbo Fang received the B.S. and M.S. degrees from The Second Artillery Engineering College, Xi’an, China, in 1994 and 2000 respectively. He is currently a Ph.D candidate in Beijing University of Technology, Beijing, China. His research interests are embedded systems, system-level design method and wireless communication.

Zhangqin Huang received Ph.D. degree in School of Electronics and Information Engineering, Xi’an Jiaotong University, Xi’an, China, in 2000. He worked as a post doctoral fellow in Eindhoven University of Technology, since 2001 through 2003. He is currently a full-time professor in Beijing University of Technology. His research interests are embedded systems and system-level design.

Yibin Hou received Ph.D. degree in Eindhoven University of Technology, Eindhoven, The Netherlands, in 1986. He is currently the vice-president of Beijing University of Technology, Beijing, China. His research interests are embedded systems and human-computer interaction.