

Wireless Protocol Verification: A Stochastic Process Algebraic Framework

Yongmei Wei and Kambiz Homayounfar

PHYBIT, INC.

AM Building 5F, 2-3-3 Higashi Gotanda, Shinagawa, Tokyo 141-0022, Japan

E-mail: kambiz@phybit.com

†PHYBIT, INC.

10-10 Cendex Center, 120 Lower Delta Road, Singapore 169208

E-mail: bijan@phybit.com

Abstract The layers 2 and 3 of mobile broadband protocols are complex specifications whose verification is notoriously hard. Informal methods for protocol verification are subject to misinterpretations and easily lead to bugs and deadlocks. Formal methods from the field of computational logic are not widely known in wireless sciences. This paper presents a formal framework for verification of wireless protocols based on Stochastic Process Algebra (SPA). We illustrate the use of SPA by taking the Mobile WiMAX as an example and show how it can apply to other standards too.

Key words Stochastic Process Algebra, WiMAX, Protocol Verification, Formal Methods.

1 Introduction

Layer 2 and 3 protocols of mobile broadband play important roles in the wireless communication systems and the overall system performance largely depends on the performance of these two layers. Unfortunately, the implementation of the protocols often leads to extremely complicated and unpredictable behavior. Generally, it takes much longer time to diagnose bugs or deadlocks after the development phase than in the early design phase. Therefore, verification and performance evaluation of the protocols are expected to be integrated into the design process in the very beginning.

Since formal methods provide a mathematically-based way to conduct protocol specification and verification, they have been developed as promising candidates to be early integrated into the development of complicated communication protocols and have been successfully applied in the development of several protocols[2][5][3]. Among all the formal methods, stochastic process algebra (SPA), with their unique features offers means to build promising framework to conduct protocol verification.

Three distinguished features make SPA a strong candidate for protocol verification. Firstly, the description language is compositional. This feature allows building and verifying highly modular and hierarchical system from smaller blocks and also allows re-usability for the smaller blocks. More importantly, there exists an operator named hiding allows one to abstract from internal details at any level of system description. Secondly, the existences of equational laws and notion of bisimilarity allow one to prove equivalence or other relations between an abstract specification and its implementa-

tions. Thirdly, temporal information has been attached to the process description in SPA. This feature allows quantitative analysis in terms of delay, throughput and determination of specific parameters. In this paper, we illustrate the use of SPA by taking the layer 2 of WiMAX system as an example. In Section 2, a short description and introduction of the notations of SPA are provided. In Section 3, a general description of a protocol verification framework by SPA is introduced. In Section 4, the framework is further explained with the layer 2 of WiMAX as an example.

2 Stochastic process algebra

2.1 Notations

In this paper, the notation generally follows performance evaluation process algebra (PEPA)[4], which is one of the most important types of SPA. In PEPA, a system is modeled as a set of processes carrying out activities either individually or in co-operation with other processes. The activity is characterized by an action type and a delay determined by an exponentially distributed random variable. In the following, a list of important operators is introduced.

Parallel $P||_{\lambda}Q$: The processes P and Q are running parallel with events belonging to the set λ being synchronized.

Interleave $P||Q$: The processes P and Q are interleaved running with no synchronization events. $||_{i=1}^{\alpha}P_i$ means that processes $P_i, 1 \leq i \leq \alpha$ are interleaved running with no synchronization events.

Prefix $Q = (a, TT).P$: Q is a process which sends

out event a after a certain delay and then behaves like the process P . The delay is characterized as a random variable with exponential probability density function. If $TT = infty$, it means that event a is a passive action which need to be synchronized. To be more specific, the same action is required to happen in another process to enable this action in process Q . If $TT = iM$, it means that the event a immediately runs during the corresponding process.

Choice $Q = (a, TT1).P + (b, TT2).R$: Q behaves either as a process which sends out event a , then behaves like the process P , or a process which sends out event b , then behaves like the process R . The choice is determined by which event happens earlier.

Hiding P/L : This process behaves like P except that any activities of types within the set L are hidden. This operator is very important because it allows the system description any level of abstractness.

2.2 Bisimulation equivalences

One of the most important feature of SPA is the existence of bisimulation equivalence[3]. This is because notion of equivalence allows validation through the comparison of system description. Below shows the definition of weakly bisimilar, which is the most practical type of equivalence.

Definition 2.2 V and Q are weakly bisimilar, which is expressed as $V \approx Q$, if they are contained in a binary relation γ on L such that each $(\hat{V}, \hat{Q}) \in \gamma$ implies for all $\alpha \in Act$:

- if $\hat{V} \xrightarrow{\alpha} \hat{V}'$ then there is \hat{Q}' such that $\hat{Q} \xrightarrow{\alpha} \hat{Q}'$ and $(\hat{V}', \hat{Q}') \in \gamma$.
- if $\hat{Q} \xrightarrow{\alpha} \hat{Q}'$ then there is \hat{V}' such that $\hat{V} \xrightarrow{\alpha} \hat{V}'$ and $(\hat{V}', \hat{Q}') \in \gamma$.

It is worth mentioning that weak bisimilarity has the property of *substitutive* with respect to the parallel operators such as \parallel and \parallel_{λ} .

This implies that, for $system = V \parallel Q$ and $system1 = \tilde{V} \parallel \tilde{Q}$,

- if $V \approx \tilde{V}$ and $Q \approx \tilde{Q}$, then $system \approx system1$ holds true.

As a result, the proof of equivalence between two large system can be compositional reduced to the that between several smaller systems. This is important because, generally, larger system leads to large state space. In a complex system, the overall state space is often too large to handle directly.

2.3 Tools

The java edition of PEPA Workbench developed by people from university of Edinburgh, is used for the deadlock and state space analysis[6] in this paper. This tool can locate the deadlock, derive the state space and do the quantitative analysis such as steady state probability. The analysis of deadlock and state space helps prove the correctness of the modeling because mistakes and/or misunderstanding occurs quite often during the process

of direct translation from the standard to either codes or any form of modeling, and it is always extremely difficult to locate the errors in a complicated system such as WiMAX. Furthermore, this tool can also operate with other tools such as Argo/UML whereby the UML state machine can be loaded and *Möbius* and PRISM to do the model checking[2].

3 General description of framework for protocol verification by SPA

There are majorly two types of techniques for the protocol verification in this paper. The first one is reachability analysis. Techniques used for reachability analysis are based on state space exploration. By conducting state space exploration, deadlock or livelocks can be detected. The second one is the equivalence checking. Bisimilarity property is used to check whether two specifications describe the same behavior. Normally, a more abstract service specification is formed and equivalence checking is realized through determining whether the service specification is bisimilarly equivalent to the functional description.

Figure 1 shows the flowchart of the proposed framework for protocol verification. The first step is doing research and analysis on the related standards, such as [1] for WiMAX. The second step is building the system functional description in an algebraic and compositional way, i.e. by using the compositional operators such as parallel, a highly modular and hierarchical system is expected to be built by isolating processes handling different type of work. Furthermore, a tree-graph describing the hierarchy of all the processes is built based on the system algebraic specification. The third step is doing reachability analysis and equivalence checking on each processes which do not have any lower level process, i.e. the leaf-process. The fourth step is doing reachability analysis and equivalence checking on the processes one level higher. This step is repeated until it reaches the top level which is the whole system specification. It is worth mentioning that simplified leaf-processes are required to be found for the verification of the higher level processes to reduce computational complexity and memory consumption.

To sum up, verification is done in a modular and hierarchical way. The reason for doing the verification in such a way is that direct combination between lower level processes leads to large state space, and thus, suffers from both high computational complexity and high consumption of memory. To save both the computational complexity and memory cost, the state spaces of each lower level processes are minimized with respect to bisimilarity before combining the lower level processes to analyze higher level processes.

Table 1: Broadcasting processes

Periodically broadcasting processes for all SSs	
Notation	Explanation
B_0	Process broadcasting UCD
B_1	Process broadcasting DCD
B_2	Process broadcasting UL-MAP
B_3	Process broadcasting DL-MAP
B_4	Process broadcasting MOB_NBR-ADV

4 Protocol verification of WiMAX base station

In this Section, the second and third steps of the framework introduced in 3 are introduced with WiMAX as an example.

4.1 Overall system modeling

In this Section, the overall description of level 2 layer protocol is presented for the base station. Since the processes in subscribe stations(SSs) are much simpler compared with those of base station(BS), they are omitted due to the space limitation. Without loss of generality, it is assumed a BS accommodates a number of α (SSs) and each SS accomdodates a number of β active service flows. The operations of BS includes both broadcasting management messages such as downlink/uplink channel descriptors to all SS, and receiving/sending management messages/data to each SS/service flows.

Based on [1], the overall process are modeled as various processes interleaved with each other shown below,

$$(|_{i=1}^{\alpha} N_i) || B \quad (1)$$

The process B deals with broadcasting messages

$$B = ((B_0 ||_{\epsilon_0} T_0) || (B_1 ||_{\epsilon_1} T_1) || (B_2 ||_{\epsilon_2} T_2) || (B_3 ||_{\epsilon_3} T_3) || (B_4 ||_{\epsilon_4} T_4)) \quad (2)$$

where T_m are the timers controlling the period of the broadcasting operations and ϵ_m represents the synchronized message set with the process for timer T_m . For example, T_0 represents the timer controlling the period of the broadcasting UCD message. Table 1 shows the definition of the processes involved in process B .

The process N_i deals with messages from the unregistered mode, it accepts messages leading to network entry process R_i and is modeled as,

$$N_i = R_i ||_{\epsilon_{i1}} T_{i1} ||_{\epsilon_{i2}} T_{i2}; \quad (3)$$

where T_{i1} and T_{i2} are two timers T9 and T17 involved in the process R_i , and ϵ_{i1} , ϵ_{i2} are the sets of the corresponding synchronized events. Table 2 shows the description of processes involved in N_i .

In process R_i , if the registration is successful, the process goes to O_i which means that the SS is in operational mode, otherwise it goes back to N_i meaning unregistered SS.

Table 2: SS-level processes

SS-level processes for the i -th SS	
Notation	Explanation
N_i	Process for unregistered mode
R_i	Process for network entry mode
RR_i	Process for network re-entry mode
O_i	Process for registered mode
P_i	Process for Periodic ranging mode
F_i	Process for SF control mode
A_i	Process for scanning mode
H_i	Process for handoff mode
I_i	Process for idle mode

For O_i , it accepts messages either leading to service flow management, periodic ranging, idle mode, handoff mode or scanning mode.

The O_i is modeled as,

$$O_i = (P_i ||_{\epsilon_{i2}} T_{i2} ||_{\epsilon_{i3}} T_{i3}) || (A_i ||_{\epsilon_{i6}} T_{i6} ||_{\epsilon_{i4}} T_{i4} ||_{\epsilon_{i5}} T_{i5}) || (H_i ||_{\epsilon_{i7}} T_{i7}) || (I_i ||_{\epsilon_{i8}} T_{i8}) || (F_i). \quad (4)$$

where $T_{i2} - T_{i8}$ and $\epsilon_{i2} - \epsilon_{i8}$ are the corresponding timers and synchronized event sets involved in the processes. Among all the processes involved in O_i , Service flow management F_i is the mostly complicated processes because one SS can have multiple service flows and each service flow can have multiple transactions. Thus, two levels of processes are required. For the upper level process, it describes the state transitions regarding to each service flow. In this level, there is also another process C_{ij} introduced to control the transaction number and send out a msg when the transaction number is zero. For the lower level processes, they describe the state transitions regarding to each transactions. With the processes from these two levels, F_i is modeled as,

$$F_i = ||_{j=1}^{\beta} (F_{ij} ||_{\eta_{ij0}} C_{ij} ||_{\eta_{ij1}} (||_{k=1}^{\gamma} (F_{ij0k}))) \quad (5)$$

where

$$F_{ij0k} = (F_{ij1k} ||_{\epsilon_{ijk0}} T_{ijk0} ||_{\epsilon_{ijk1}} T_{ijk1}) || (F_{ij2k} ||_{\epsilon_{ijk2}} T_{ijk2} ||_{\epsilon_{ijk3}} T_{ijk3}) || (F_{ij3k} ||_{\epsilon_{ijk4}} T_{ijk4} ||_{\epsilon_{ijk5}} T_{ijk5}); (F_{ij4k} ||_{\epsilon_{ijk6}} T_{ijk6} ||_{\epsilon_{ijk7}} T_{ijk7}) || (F_{ij5k} ||_{\epsilon_{ijk8}} T_{ijk8} ||_{\epsilon_{ijk9}} T_{ijk9}) || (F_{ij6k} ||_{\epsilon_{ijkA}} T_{ijkA}); \quad (6)$$

where η_{ij0} represents the synchronized msg set between C_{ij} and F_{ij} and η_{ij1} represents the synchronized message set between F_{ij} , C_{ij} and F_{ij0k} . Table 3 shows the description of the processes involved in F_{ij0k} .

After carefully examining the overall system specification shown in equation (1), for all the processes involved in the operation of a base station (BS), it can be seen that they fall into processes of four levels. Figure 2 shows the corresponding tree-graph of the processes. In the top level, the overall process is modeled as the broadcasting process paralleled run with processes controlling

Table 3: Transaction-level processes

For the k th transaction of the j th SF in i th SS	
Notation	Explanation
F_{ij1k}	DSA transactions
F_{ij2k}	DSC transactions
F_{ij3k}	DSD transactions
F_{ij4k}	Remote DSA transactions
F_{ij5k}	Remote DSC transactions
F_{ij6k}	Remote DSD transactions

each SSs. For the second level, the broadcasting process can be further modeled as four level-2 broadcasting process. The processes controlling each SSs, however, is more complicated and can be further modeled as R_i being the entry process. Since R_i, RR_i, F_i, H_i, A_i and I_i can be changed between themselves, all these processes belong to level-2 processes. Among these level-2 processes, F_i can be further modeled by level-3 processes dealing with different service flows. For level-3 processes, they are further modeled by level-4 processes dealing with different transactions. In the next two sub Sections, the broadcasting processes is selected as an example due to its simplicity, to illustrate how hiding and bisimilarity help to reduce the complexity of state space exploration.

4.2 Verification of leaf-process

For the leaf-process, the process B_0 is taken as an example. Once BS starts to work, it begins to send the UCD msg ms_{00} and start the timer T_0 by sending a_{01} immediately. Then, the process behaves as B_{00} . B_{00} waits for the timeout event a_{00} from the process for timer T_0 running independently. After receiving a_{00} , it sends the msg ms_{00} and starts the timer by sending a_{01} again. The process repeats afterwards. This whole procedure is represented as,

$$B_0 = (ms_{00}, TT0).(a_{01}, iM).B_{00}; \quad (7)$$

$$B_{00} = (a_{00}, infty).(ms_{00}, iM).(a_{01}, iM).B_{00}; \quad (8)$$

The process of timer T_0 is modeled as,

$$T_0 = (a_0, infty).T_1; \quad (9)$$

$$T_1 = (a_1, t).T_0. \quad (10)$$

T_0 is the starting point. It waits for the start timer msg a_0 to start the timer and then behaves like T_1 . After a certain delay decided by t , T_1 sends the timeout msg a_1 and behaves like T_0 again.

The following subsystem is used for the deadlock and state space analysis of the modeled processes,

$$(B_0 ||_{\epsilon_0} T_0) \quad (11)$$

By using the PEPA Workbench, it shows that the subsystem above contains no deadlocks. Total number of states is 4, total number of transitions is 4 and it takes

Table 4: Number of states and transitions for each leaf-process

Process	States	Transitions
R_i	669	1650
RR_i	165	434
I_i	2191	5241
H_i	214	487
A_i	1067	2578

0.171 seconds to complete the derivation of the state space.

It is worth mentioning that deadlock and state space analysis of the leaf-process not belonging to broadcasting processes are different from those belonging to broadcasting. This is because, broadcasting processes do not wait for the response from SSs. The rest leaf-processes, however, require communication with SSs. Thus, processes for the corresponding SS and medium are required for the verification process. If it is assumed that RS_i and $Medium$ refer to the processes for SS and medium, respectively, the subsystem required is

$$(R_i ||_{\epsilon_{i1}} T_{i1} ||_{\epsilon_{i2}} T_{i2}) ||_{\epsilon_{iM}} Medium ||_{\epsilon_{iS}} RS_i \quad (12)$$

where ϵ_{iM} and ϵ_{iS} represents the events required to be synchronized between BS and medium, and between medium and SS, respectively.

Table 4 shows the number of states and transitions for most of the leaf-process. It can be seen that idle mode has the most number of states and transitions.

4.3 Verification of non-leaf-process

To verify non-leaf-process B , the processes B_1 - B_4 are represented just like B_0 as,

$$B_1 = ((ms_{01}, TT1).(a_{03}, iM).B_{01}); \quad (13)$$

$$B_{01} = (a_{02}, infty).(ms_{01}, iM).(a_{03}, iM).B_{01}; \quad (14)$$

$$B_2 = ((ms_{02}, TT2).(a_{05}, iM).B_{02}); \quad (15)$$

$$B_{02} = (a_{04}, infty).(ms_{02}, iM).(a_{05}, iM).B_{02}; \quad (16)$$

$$B_3 = ((ms_{03}, TT3).(a_{07}, iM).B_{03}); \quad (17)$$

$$B_{03} = (a_{06}, infty).(ms_{03}, iM).(a_{07}, iM).B_{03}; \quad (18)$$

$$B_4 = ((ms_{04}, TT4).(a_{09}, iM).B_{04}); \quad (19)$$

$$B_{04} = (a_{08}, infty).(ms_{04}, iM).(a_{09}, iM).B_{04}. \quad (20)$$

where $ms_{01}, ms_{02}, ms_{03}$ and ms_{04} represent the actions sending DCD, UL-MAP, DL-MAP and MOB_NBR-ADV messages, respectively, a_{02}, a_{04}, a_{06} and a_{08} represent timeout events for the corresponding timers, and a_{03}, a_{05}, a_{07} and a_{09} represent the events to start the corresponding timers.

If no hiding or bisimilarity property are used, the following subsystem is used for the deadlock and state space analysis of the modeled processes,

$$((B_0 ||_{\epsilon_0} T_0) || (B_1 ||_{\epsilon_1} T_1) || (B_2 ||_{\epsilon_2} T_2) || (B_3 ||_{\epsilon_3} T_3) || (B_4 ||_{\epsilon_4} T_4)). \quad (21)$$

By using the PEPA Workbench, it shows that the subsystem above contains no deadlocks. Total number of states is 1024, total number of transitions is 5120 and it takes 5.281 seconds to complete the derivation of the state space.

Practically, the events related with the timer can be seen as internal events and the hiding operator can be used to reduce the total number of states and transitions. For example, if the events related with T_0 is hidden, i.e. the process is modeled as

$$\begin{aligned} & ((B_0||_{\epsilon_0}T_0)/\{\epsilon_0\})||((B_1||_{\epsilon_1}T_1)||((B_2||_{\epsilon_2}T_2) \\ & ||((B_3||_{\epsilon_3}T_3)||((B_4||_{\epsilon_4}T_4))). \end{aligned} \quad (22)$$

Total number of state is reduced from 1024 to 768, total number of transitions is reduced from 5120 to 3584. We can see that 25 percent savings have been achieved in terms of state space. The state space can be further decreased with more events related with timers are hidden.

5 Conclusion and future work

This paper propose a framework of protocol verification using SPA. Generally speaking, the framework follows a compositional way to use the algebraic achievements from SPA to achieve verification in a more efficient way. Furthermore, the framework has sound mathematical basis and provides reliable results compared with informal methods and simulations. It is worth mentioning that this paper does not intend to provide a thorough detailed verification for layer-2 of WiMAX, but to propose a methodology. Whole system verification and optimal parameter determination by the quantitative analysis through SPA, will be conducted in the future.

References

- [1] *IEEE Std 802.16e-2005 and IEEE Std 802.16-2004/Cor 1-2005*. IEEE, 2005.
- [2] Stephen Gilmore and L. Kloul. A unified tool for performance modelling and prediction. *Reliability engineering & systems safety*, 89:17–32, 2005.
- [3] H. Hermanns, U. Herzog, and V. Mertsiotakis. Stochastic process algebras between lotos and markov chains. *Computer Networks and ISDN Systems*, 30(9–10):901–924, 1998.
- [4] Jane Hillston and Marina Ribaudo. Stochastic process algebras: A new approach to performance modeling. In *Modeling and Simulation of Advanced Computer Systems*, pages 235–256. Gordon Breach, 1998.
- [5] K.N.Sridhar and C. Gabriel. Describing ieee 802.11 wireless mechanisms by using the π -calculus and performance evaluation process algebra. *Lecture notes in computer science*, 3236:233–247, 2004.

- [6] S.Gilmore and J. Hillston. The pepa workbench: A tool to support a process algebra-based approach to performance modelling. *Lecture Notes in Computer Science*, 794:353–368, 1994.

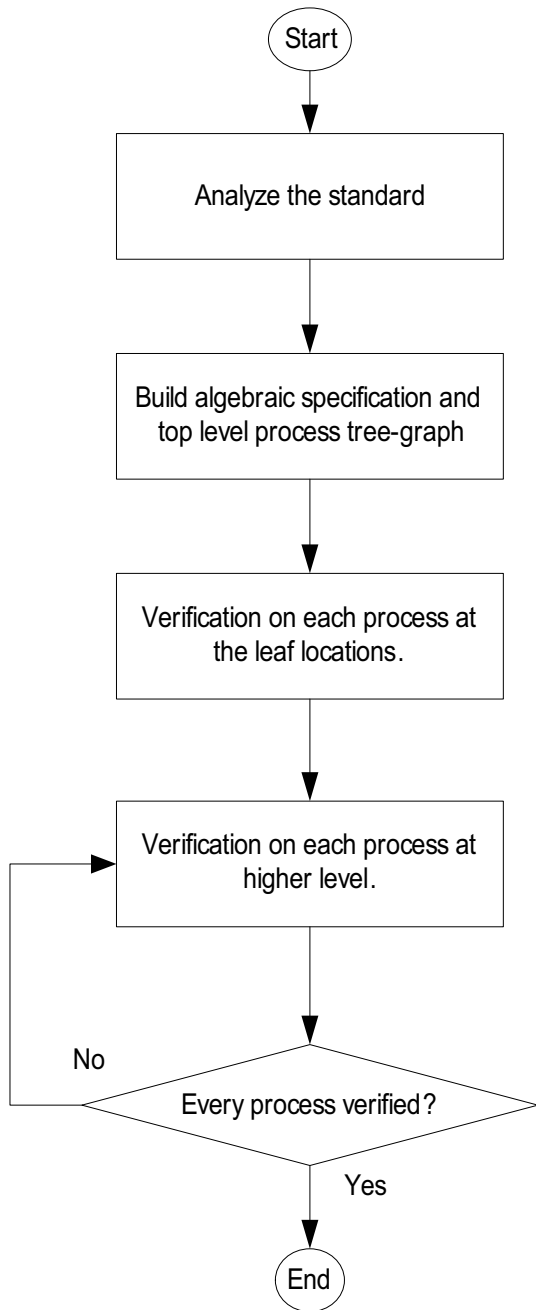


Figure 1: The framework of protocol verification by SPA

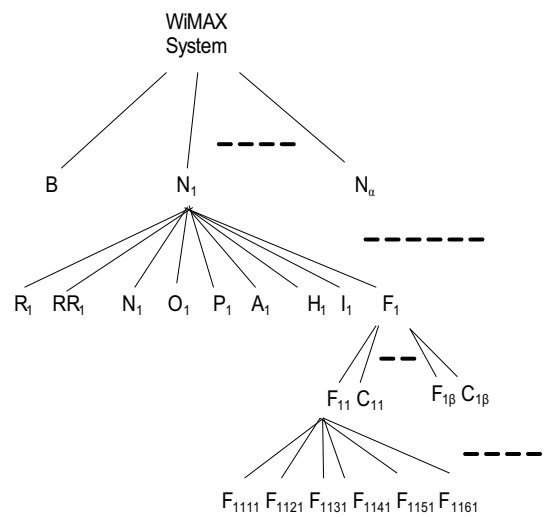


Figure 2: Tree graph of the processes of WiMAX