Towards Formal Modeling and Analysis of BitTorrent using Coloured Petri Nets

Jing LIU[1, 3], Xinming YE[2], Tao SUN[2]

[1] Institute of Computing Technology, Chinese Academy of Sciences, China
[2] College of Computer Science, Inner Mongolia University, Hohhot, China
[3] Graduate University of Chinese Academy of Sciences, China
liujing@ict.ac.cn, {xmy, cssunt}@imu.edu.cn

Abstract. BitTorrent is widely adopted in P2P applications, such as file sharing and video streaming. As intricate communication and concurrency are characteristics of BitTorrent, it is difficult to formally and effectively model its functional behaviors in peer-level. In this paper, a coloured Petri Nets based hierarchical modeling architecture and detailed model instances of BitTorrent are proposed. Then simulation, state spaces analysis and model checking technologies are utilized combinatively to validate the formal models and verify the functional properties of BitTorrent. The proposed formal model could not only be served as an unambiguous and visual formal specification, but also facilitate the behaviors simulation and properties verification where it relieves the notorious state space explosion problem.

Keywords: BitTorrent, coloured Petri Nets, simulation, state space analysis

1 Introduction

Various peer-to-peer (P2P) applications have become prodigiously popular on the Internet. BitTorrent [1] accounts for more than a half traffic of all P2P traffic in most countries, so it is widely recognized as a more popular P2P content distribution protocol. The advantages of BitTorrent are high scalability, stable distribution performance and easy deployment, all of which result from the basic idea of BitTorrent, that is, individual peers could effectively utilize their incoming access link bandwidth. Peers participate in an application level overlay network and behave as both client and server. They download and upload file pieces mutually at the same time. This kind of cooperative behavior makes the file sharing more effective and more efficient. However, it also introduces more intricate communication and concurrence, making the functional behaviors modeling and analysis of BitTorrent more difficult.

In recent years, there are many notable studies concerning about different aspects of BitTorrent, such as system design, traffic measurement, performance analysis and key algorithms optimization. As for the modeling of BitTorrent, most researches focus on the performance modeling and analysis. [2] utilizes a simple fluid model to describe and analyze the dynamic behaviors of the BitTorrent systems. [3] utilizes a Markov model to analyze the freerider phenomenon, where a peer just downloads from others without uploading its contents as expected. [4] extends the fluid model

mentioned in [2] to perform extensive measurement and trace analysis of a single-torrent system. [5] adopts a simple mathematical model to characterize the group-level properties of BitTorrent system execution and perform a complex observed performance analysis. [6] models the whole downloading process as several consequent phases using a three-dimension Markov chain, which is suitable for capturing the peer downloading behaviors compared with real world traces. It also launches notable analysis of the stability of BitTorrent. [7] uses a stochastic fluid model to characterize the behavior of on-demand stored media content delivery based on BitTorrent. It provides insight into transient and steady-state system behavior for its performance evaluation.

To summarize, most of the studies adopt various mathematical models to analyze the performance of BitTorrent, and they usually focus on the aggregate properties, such as average downloading or uploading rates, network utilization and cost, etc. Few studies focus on the functional behavior modeling in peer level, which aims to construct a formal function model of BitTorrent and validate its soundness.

Therefore in this paper, a coloured Petri Nets based function model of BitTorrent system is proposed and effectively validated. Coloured Petri Nets are quite suitable for modeling and validating the system in which concurrence, communication and synchronization play a major role. Using coloured Petri Nets to model BitTorrent system is absolutely a good choice, because these models not only specify the functional details hierarchically and unambiguously, but also present visible execution of the concurrent behaviors of BitTorrent. Our contributions are listed as follows:

+ *A modeling architecture of BitTorrent is proposed*. It presents significant guidance about model hierarchy, data abstraction and model refinement. It also applies to modeling other protocols with intricate communication and concurrence as their behavior characteristic like BitTorrent.
+ *A coloured Petri Nets based hierarchical model of BitTorrent is constructed*. To the best of our knowledge, it is the first time to present a function model of BitTorrent in peer level. It could not only be served as an unambiguous and visual formal specification for different system implementations, but also facilitate the behaviors simulation and properties verification of BitTorrent.
+ *An effective model validation and analysis method is presented*. Taking full advantage of CPN Tools, an integrated validation and analysis method is performed, combining simulation, state space analysis and model checking technologies. They are used towards different abstract levels of above models to validate the model soundness and effectiveness, and check whether those models satisfy the key requirement properties of BitTorrent system, such as no out-of-orders executions, or random downloading behaviors, etc.

The rest of this paper is organized as follows. As the background of our studies, section 2 presents an overview of BitTorrent system and coloured Petri Nets with CPN Tools. Then, section 3 describes the modeling architecture as a general guidance, and specific model instances are constructed and explained in section 4, together with some modeling assumptions and data modeling. Section 5 focuses on model validation and analysis, which is used to confirm the validity of formal models proposed in section 4. Finally, section 6 concludes the paper and sketches our future research issues.

2 Background

2.1 BitTorrent Overview

BitTorrent aims to facilitate fast downloading of popular files. According to the informal specification of BitTorrent [1], a sharing file is divided into several pieces of fixed size (e.g. 256 KB each), which are basic sharing units during file distribution. A special torrent file, as the uniform identification of the source file, is made to record the information and data hash of these pieces. All participating peers that download or upload the same file will form a random and temporary mesh network, and they download different pieces from different peers. If there are numerous simultaneous peers sharing the same file, downloading an entire copy will become faster.

In a BitTorrent distribution network, there are two kinds of peers: *leecher*, who downloads its lack and uploads its having at the same time, and *seed*, who has entire file and just uploads for other leechers. Besides, there is a *tracker* to keep tracking information (e.g. peer positions and their downloading progress) of all the participating peers dealing with the same file. It stores and maintains such information, and uses them to organize the peer list, which helps a new leecher find peer candidates to connect. As illustrated in figure 1, a leecher firstly asks the tracker for the peer list. Then it establishes connections to those peers in the list which are sharing the same file, and records which pieces they have through bitmap messages. Finally, this leecher requests pieces from other connecting peers, leechers or seeds, and continues the file sharing until it becomes a seed. Many protocols and algorithms play a significant part in above procedures.

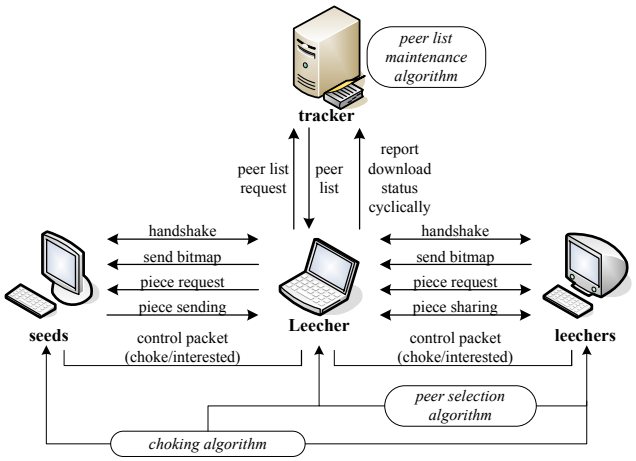


Fig. 1: The major components of a BitTorrent system.

There are two specific protocols in BitTorrent. One protocol specifies the communication between a peer and a tracker, called *tracker protocol*; the other specifies the interaction behaviors between peers, called *peer protocol*. The tracker protocol describes the functionalities including peer list request and response, and peer downloading status cyclical report. It is much simpler compared with the peer proto-

col. The peer protocol describes various interactions during file sharing between a leecher and a seed, or between leechers. It consists of handshake procedure, bitmap exchange, piece request and sending, piece having announcement, and other control communication used to achieve incentive mechanism [1] in BitTorrent.

Besides, there are several key algorithms to promote the efficiency of file sharing, including peer list maintenance algorithm implemented in trackers, and piece selection algorithm and choking algorithm implemented in peers. As described in [1], there are four default piece selection algorithms, i.e. strict priority, rarest first, random first piece and endgame mode. They work collaboratively to improve the whole distribution performance of a BitTorrent system. Choking algorithm is designed to guarantee reasonable and fair downloading speed for every leecher, and put some positive feedback to freeriders.

In practice, a BitTorrent system is very complicated. A peer usually participates in different file downloading and uploading procedures. Learning from the good modeling exercise, we tradeoff between the sufficiency and validity of formal models and simplicity and feasibility of the model analysis. We make full use of hierarchical modeling capability of coloured Petri Nets to model the BitTorrent protocol in several abstract levels, which does not lost the necessary functional details of BitTorrent, together with a modest size of the model for practical analysis.

2.2 CPN and CPN Tools

Coloured Petri Nets (CPN) [8] is always adopted to model and validate systems with high concurrence and complex communication. In recent years, there are many successful projects [11, 12, 13, 14]. CPN has many strong capabilities used to facilitate modeling complex systems, i.e. token colors with abundant data types and operations; hierarchical modeling using substitution transitions, page instances and fusion sets; flexible functional programming language to specify control and data constrains. Such techniques facilitate modeling the BitTorrent into several abstract layers, and expressing data type and functional behaviors accurately and flexibly.

These are several CPN based modeling tools for researcher to do experiments. As far as well known, CPN Tools [9, 10] is the most powerful one. Regarded as industrial-strength software for modeling and analysis CPN models, CPN Tools gains more than 8000 users from nearly 140 countries, and widely adopted in many significant projects [10]. In the paper, we take full advantage of editing, simulation, and state space analysis capabilities of CPN Tools to construct and validate BitTorrent models.

3 Modeling Architecture

The informal specification of BitTorrent [1] just covers the fundamental and necessary parts, such as systems deployment, key data structures, core algorithms, and main flow of contents publishing and files distribution. From the point of view of functional modeling, there are two major hurdles in constructing an accurate and appropriate model based on such specification. On the one hand, some sections in the specification do not need to be modeled, such as system deployment procedure or

some data collection behaviors for user layer displaying. Modeling these behaviors contributes few to system functional analysis, and to make matters worse, introduces incogitable but unnecessary state space explosion. How to distinguish such kind of functionalities from other indispensable parts is a really challenge. On the other hand, some detailed algorithms or message interactions in the specification are not explained clearly or even not mentioned at all. Take the choking and interesting messages for example, the trigger time and orders of such messages interaction are not mentioned clearly. It needs further consideration and complementarities from the perspective of design or implementation phases.

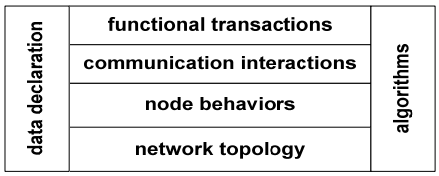


Fig. 2: The modeling architecture of BitTorrent.

In this paper, we propose a non-trivial modeling architecture to cope with above two challenges. As illustrated in figure 2, the modeling architecture contains four layers, which represent different modeling levels, together with data declaration and algorithms layers throughout. We could firstly mode the network topology, and then refine the node behaviors and other upper layer issues, as this paper demonstrated. With the equivalent effects, we could also model detail algorithms and specific functionalities of peer entities firstly, and then compose them to form communication interactions and construct node and network layers. Each layer focuses on certain aspects of system functional behaviors, and the tradeoff between the model size in one layer and the function sufficiency in that layer should be paid more attentions.

Network topology layer focuses on the modeling of entire network environments, including the participating entities and their relationship from the network topology point of view. Especially, the number of different types of entities and their position in the network environments should be considered carefully. Redundant entities or incorrect relations not only introduce a potential huge state space, but also could not give prominence to the key properties of the system.

Node behaviors layer focuses on the execution states and their transfer relation in a specific entity, such as a tracker or a peer node. As for network protocols, sending requests and receiving responses, together with some connectivity control actions are usually modeled in this layer.

Communication interactions layer focuses on messages interactions between protocol entities. As for network protocols, collecting property data, generating requests, parsing response and switching to subsequent processing are major modeling issues in this layer. The logical relationship of such behaviors needs careful consideration.

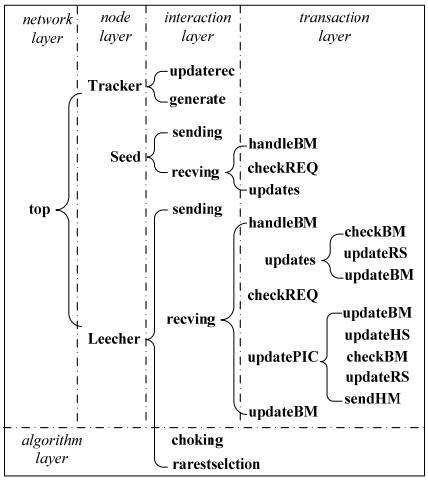
Transactions and algorithms layer focuses on the detailed functionalities, for example, the control flows, maintenance of key data structures, sampling the required data, and core algorithms. The redundant or inaccurate modeling in this layer will lead to notorious state space explosion, especially when the high concurrence and complex communications exist. Therefore, we should iteratively refine the models to construct an optimum model with modest sizes and functional descriptions in different layers.

The model size of this layer is often bigger than that of other layers, so we could divide transaction layer into several sub-layers for legible modeling.

To sum up, taking full advantage of hierarchical abstraction methodology, above modeling architecture facilitates modeling system functionalities into several abstract layers, and expressing behavior details accurately and flexibly. It is quite suitable and feasible for guiding complex system modeling. According to different modeling and analysis purposes, we could adjust the modeling scale inter-layer and inner-layer, and perform efficient analysis in suitable layers. CPN is considered to be an effective actualization of above modeling architecture, and the following sections demonstrate the validity of such actualization.

4 CPN Modeling of BitTorrent

Guided by the modeling architecture in above section, we construct an entire BitTorrent CPN model with 44 page instances (24 if replicated page instances are not counted), as shown in figure 3. This model assumes the absence of exceptions, that is, communication infrastructure is reliable, and there are no vulnerabilities during the protocol execution. Section 4.1 discusses some function related assumptions in modeling. Key date types are modeled as different color sets in section 4.2. From section 4.3 to 4.7, we present BitTorrent CPN models in different layers respectively. Six specific page instances are presented as a representative. They cover all modeling layers, and the most significant functionalities of BitTorrent.



pages	descriptions
top	describe the entire network topology
Tracker	describe the genaral behavior of a tracker
updaterec	update tracking information
generate	generate and send response packet with required peer list
Seed	describe the genaral behavior of a seed peer
sending	send communication packets of peer protocols
recving	receive and phrase communication packets
handleBM	describe behaviors as receiving bitmap message
checkREQ	generate a new piece packet as receiving request packet
updates	describe behaviors as receiving choking related messages
Leecher	describe the genaral behavior of a leecher peer
sending	send communication packets of tracker or peer protocols
recving	receive and phrase communication packets
handleBM	describe behaviors as receiving bitmap message
updates	describe behaviors as receiving choking related messages
checkBM	look up the required peer bitmap information
updateRS	generate a new request packet
updateBM	update bitmap information
checkREQ	generate a new piece packet as receiving request packet
updatePIC	describe consequent behaviors as receiving piece packet
updateHS	record the required piece have been stored
sendHM	generate and send have packets
choking	implement choking algorithm
rarestselection	implement the rarest piece selection algorithm

Fig. 3: The entire CPN model of BitTorrent.

4.1 Modeling Assumptions

As discussed in section 3, modeling all aspects specified in original BitTorrent specification [1] will definitely fell into the state space explosion embarrassment. There-

fore, some function related modeling assumptions should be made at first, that is, some complex behaviors should be simplified and some inconsiderable parts should be omitted directly. According to the modeling architecture, we discard the dross and select the essential, so detailed modeling assumptions are listed as follows.

+ *Some inconsiderable functionality is omitted.* Firstly, web server related processing is beyond the core BitTorrent functionalities, and not worth while to analysis. Then, some over-detailed methods, such as Bencoding and Hash checking [1], are modeled just as a transition under the reasonable assumption that they go wrong in a very small probability. At last, the hash value of torrent file is used to indentify each sharing file instead of using the whole torrent file. The generating and parsing of torrent file are omitted together.
+ *Only single file sharing is considered.* The single file sharing scenario covers all functionalities of BitTorrent, and is more feasible for analysis, so without losing the generality, we carry out simulation and state space analysis based on single file sharing configuration in BitTorrent executing.
+ *File piece is the basic sharing unit.* According to the original specification, the basic request unit of a file is slice [1], and a piece is composed of several slices. Taking into account the similar processing behaviors, we adopt piece as basic sharing unit for simpler analysis.
+ *Some less important mechanisms are simplified.* Endgame mode [1] is just a piece selection optimized method in the end phase of file downloading. Not modeling its behaviors does not affect the main functionalities of BitTorrent, and avoid introducing huge concurrent state space. Besides, as for the choking algorithm, we just model the fundamental part without optimistic unchoking and anti-snubbing [1], which are used for improving the performance and fairness of BitTorrent.

To sum up, above modeling assumptions are necessary to focus our modeling issues on the most significant parts of BitTorrent, and control the size of CPN models for effective and efficient behaviors analysis.

4.2 Data Modeling

The major data types used in BitTorrent are modeled as different color sets, shown in figure 4. We utilize simple color sets (unit, integer, and string) and compound color sets (product, record, union, and list) together to model property fields (e.g. infohash, peerid, and some flags), communication packets (e.g. handshake packets, choking packets, bitmap packets, request packets, piece packets, have packets) and key data structures (e.g. peer bitmap, piece request set, piece having set) of BitTorrent system.

It is well-known that complex color sets will possibly result in more difficult analysis work. We hold the following principle in data modeling: capturing the indispensable data elements and organizing them using suitable color sets to achieve both clear representation and easy operation. For example, the fields in communication packets used in tracker protocol and peer protocol are less than the original BitTorrent specification, because only necessary fields are picked up, and organized with suitable color sets as simple as possible.

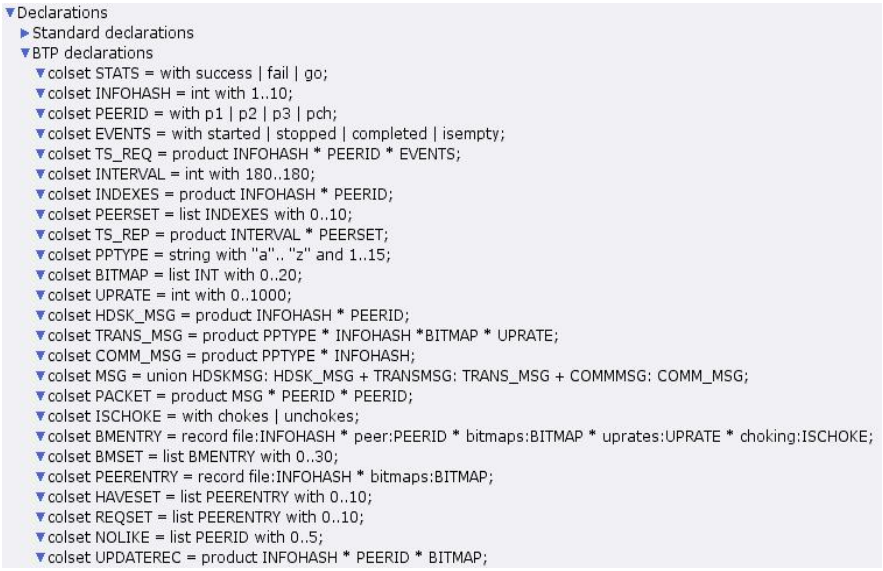


Fig. 4: The color sets in CPN model of BitTorrent.

HSDK_MSG stands for main data fields of handshake messages. TRANS_MSG stands for main data fields of bitmap, request, piece and have messages. COMM_ MSG stands for choke, unchoke, interested and uninterested messages. Union type is used to uniformly model such messages, and combined with source peerid and destination peerid to compose the entire communication packets. Besides, there are three significant data structure: BMSET, representing the piece distribution information of other peers, HAVESET, representing the index of pieces that have been downloaded, REQSET, representing the index of pieces that have been requested but not downloaded. They all modeled as list type for easy data retrieval and update. Furthermore, corresponding variables are named almost the same with its host color sets except for spelling with lowercase, so we do not list them for limited space.

4.3 Network-layer Modeling

Figure 5 indicates the top page of BitTorrent CPN models. It describes the network topology of the BitTorrent system for our analysis. There are one tracker (*Tracker*), one seed (*Seed*), and two leechers (*Leecher1 and Leecher2*). *Leecher1* acts as a new joining peer, and *Leecher2* acts as an existing leecher with part file. Because they have same behaviors, the subpages derived from the substitution transitions Leecher1 and Leecher2 in the top page are the same. We only assign different initial markings of BMSET to identify that *Leecher1* has empty file and *Leecher2* has part file, and *Leecher1* could download the file from both *Seed* and *Leecher2*. The underlying transmission network is model as several places with color set PACKET. These four entities compose a least topology set which could cover the whole desired functionalities of BitTorrent, and the protocol executions among these entities are already very complicated for feasible and effective model analysis processing.

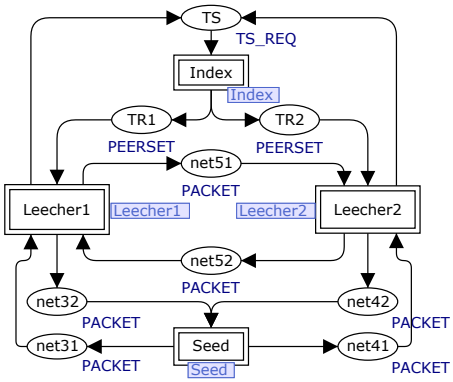


Fig. 5: The network topology model.

4.4 Node-layer Modeling

We choose the leecher model as a representative of node layer models because the tracker model is simpler and the seed model is just a bridged edition of leecher model without piece requesting related functionalities. Figure 6 indicates the specific behaviors of a leecher. The key aspect of modeling in this layer is to outline the protocol execution flows as a whole. For example, the main body of leecher behaviors is various communications involved in the tracker protocol and the peer protocol, together with their temporal orders. According to this principle, we model four kinds of leecher behaviors: peer list request, peer list parsing, packets generation and sending of the peer protocol, packets receiving and parsing of the peer protocol. These behaviors interact causally and cooperatively.

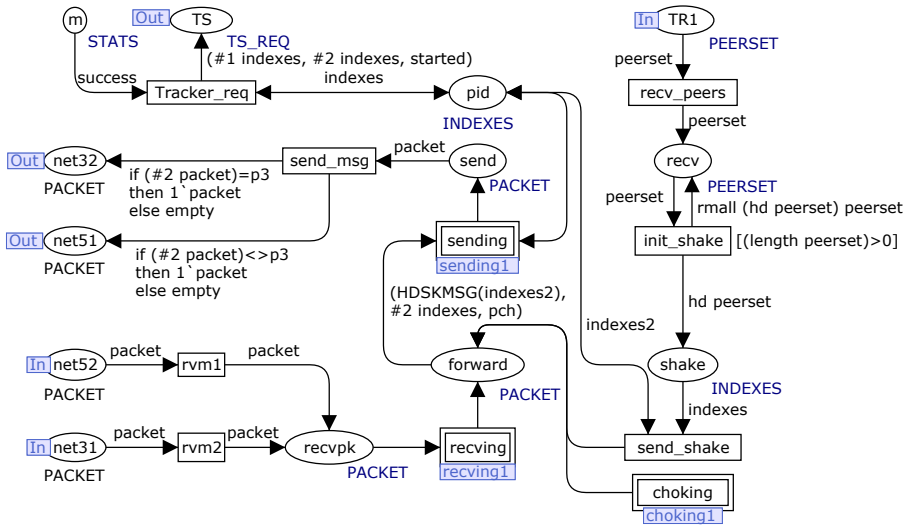


Fig. 6: The model of a leecher.

4.5 Interaction-layer Modeling

Modeling in the interaction layer focuses on the processing of protocol packets. As for the tracker protocol and peer protocol in BitTorrent, generating requests from data fields, parsing responses and switching to subsequent processing respectively are major modeling issues. As a typical example, figure 7 presents the behaviors when a leecher receives different packets from other connected peers. It is actually a subpage of substitution transitions *receive* in figure 6. On receiving handshake packets, the leecher will generate the corresponding bitmap message after successful handshake verification. On receiving (un)choke or (un)interested packets, the leecher will continue subsequent steps, which are modeled as another substitution transitions *updates* in detail. On receiving bitmap, request, piece or have packets, the leecher will carry out respective processing actions modeled as different substitution transitions.

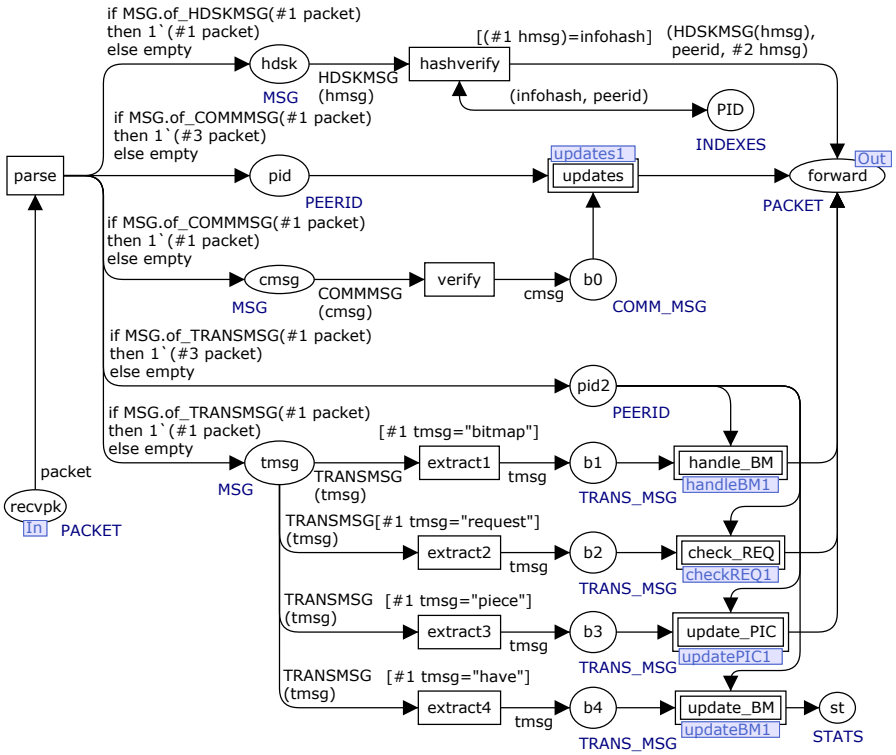


Fig. 7: The receiving behavior model of a leecher.

Detailed processing of choking and interesting related messages (COMM_MSG) is modeled with an identical substitution transition, because processing methods to these messages are similar and simple. But detailed processing of bitmap, request, piece and have messages (TRANS_MSG) is modeled with respective substitution transitions, because processing methods to these messages are more complex and different with each other. So adjusting the model scale inter-layer and inner-layer is quite helpful to obtain the modest model size for feasible analysis.

4.6 Transaction-layer Modeling

Transaction layer are fundamental page instances to model specific functionalities of BitTorrent. Modeling in this layer requires many tradeoffs to pursue the golden section of modest model size, so iterative model refinement is indispensable and significant. Two kinds of transaction pages are exemplified as follows. Figure 8 presents the behaviors when a leecher receives choking or interesting packets. It contains some substitution transitions to describe more detailed behaviors, while figure 9 is an absolute leaf page instance with no substitution transitions. It indicates the behaviors that a leecher firstly checks and calculates which piece should be requested and then generates request message with available data.

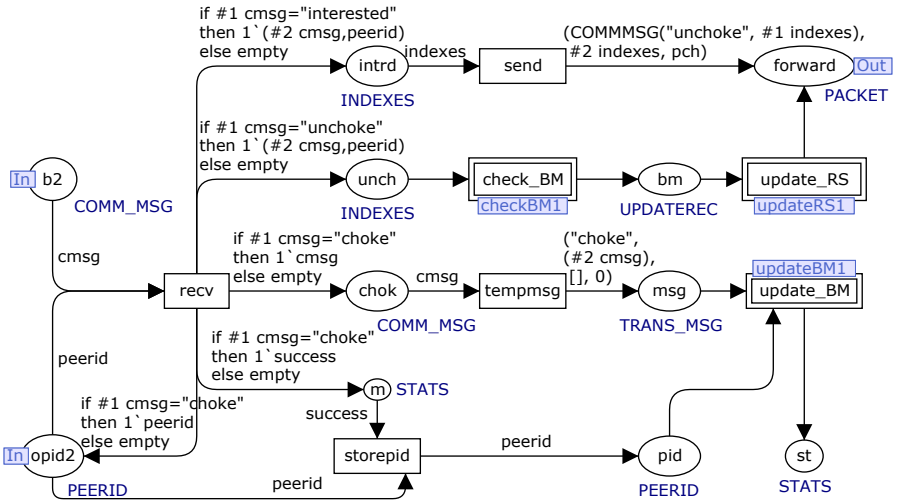


Fig. 8: The transaction behavior model of a leecher.

The size of page instances in this layer tends to be large, and any redundant or inaccurate behaviors modeling will lead to serious state space explosion. Based on our experience, two kinds of problems are worth notice. One is about concurrent operations semantics. There often exist some seeming concurrent actions, which could be modeled sequentially without any harm to protocol functionalities. For example, when a leecher has received a piece, it should update HAVESET structure and request a new piece. These two behaviors are independent, and could execute concurrently or sequentially. If model them as concurrently execution, many unnecessary concurrent states will be introduced, so we coercively arrange the execution order of these actions, that is, a control place is added to make corresponding transitions fired sequentially. The other problem is about the balance between the complexity of the net structure and the data inscriptions. It is wheezy but vital. Considering CPN Tools provides powerful ML programming language for describing constrains, we prefer specifying ML inscriptions to introducing new places or transitions when modeling some exception behaviors. Taking retrieving list data as an example, transition guard inscription is used to model null-list checking instead of making a new transition. The former does no harm to protocol functionalities, and reduces much redundant states.

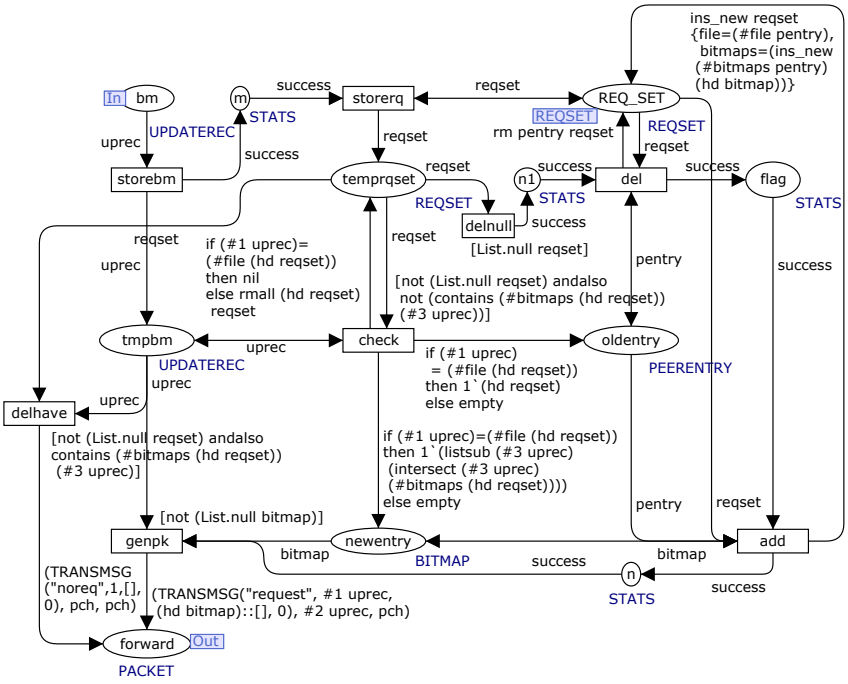


Fig. 9: The transaction behavior model of a leecher.

4.7 Algorithm-layer Modeling

There are several algorithms implemented in BitTorrent. Choking algorithm and piece selection algorithms are most important.

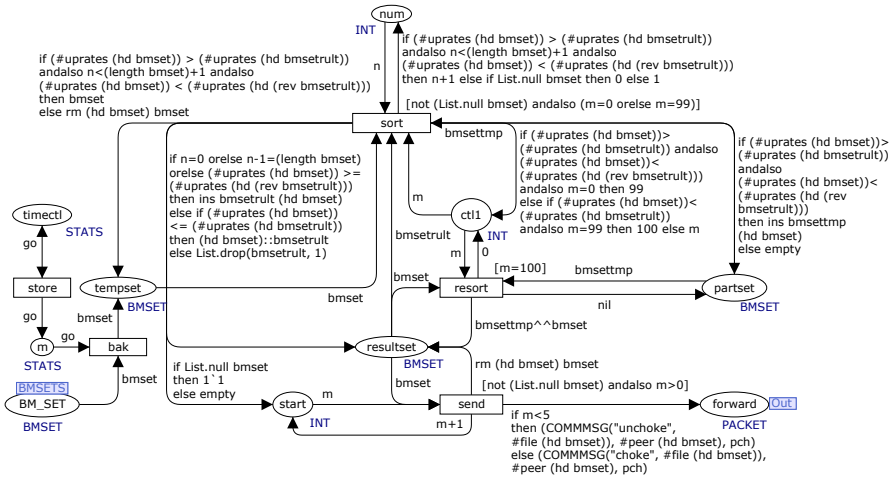


Fig. 10: The choking algorithm model of a peer.

BitTorrent has no central resource allocation, and each peer tries to maximize its own download rate based on local conditions. A peer uploads pieces to certain leechers according to the download rate they obtain from such leechers. It is a variant of tit-for-tat. In order to achieve better performance, a peer usually chokes some non-active peers temporarily, and readjusts the choking peer list periodically. The choking algorithm specified in BitTorrent is composed of three parts: the basic choking algorithm, the optimistic unchoking algorithm and anti-snubbing algorithm. In our CPN models, only the basic choking algorithms are modeled. As shown in figure 10, the main behavior in the algorithm is to order the entries in BMSET according to the download rates. The first four are considered as unchoking peers and others as choking peers, and corresponding choke or unchoke packets are sent respectively.

Piece selection algorithms focus on selecting pieces to download with modest orders for better performance. There are four piece selection algorithms. *Strict Priority* is concerned about slice downloading, that is, once a slice has requested from one peer, the remaining slices in that piece are also requested from the same peer. We do not model this algorithm because we just model the piece level behaviors as explained in section 4.1. *Rarest First* is the core algorithm in piece selection. Considering all connecting peers with a certain peer, if a piece has least copies storing among these peers, this piece should be downloaded firstly. This algorithm guarantees that the rarest pieces could be distributed as quickly and early as possible. Similar to choking algorithm, the main behavior of rarest first algorithm is to order the entries in BMSET according to the number of each piece storing in other connecting peers, and the least pieces are put forward. Because of behavior similarity to choking algorithm, models of this algorithm are not presented for space limitation. *Random First Piece* is used when downloading starts for obtaining a complete piece as soon as possible. We utilize a random initial marking to model this algorithm. At last, *Endgame Mode* is used to conquer the problem where the last piece is usually hard to get. As mentioned in section 4.1, this algorithm is not modeled because of serious concurrent behaviors.

In our constructed CPN models, we trigger the choking algorithm at the time that piece request starts. Also, we consider the executing of algorithm as atomic events, that is, the piece request procedure will never start unless the choking algorithm is over. Without losing the generality, this simplification could reduce huge concurrent behaviors and make protocol analysis more practicable. But in fact, the choking algorithm are essentially time-driven, that is, it works periodically and independently, so we will try to utilize time modeling capability provided by CPN Tools to refine algorithm models in further research issues.

5 Analysis of BitTorrent CPN Models

Having constructed the CPN models of BitTorrent, we should make further analysis to validate and revise the model, that is, to validate the effectiveness of models, and check whether those models satisfy the key requirement properties of BitTorrent system, such as no out-of-orders executions, or random downloading behaviors, etc. Unfortunately, as concurrence and intricate communication are essential characteristics of BitTorrent systems, the constructed models are so large that the direct state spaces

analysis becomes infeasible because of the notorious state space explosion problem. In order to launch practical analysis, and make it as complete and reliable as possible, we introduce an integrated method which combines CPN Tools supported simulation, state space analysis and model checking technologies, and uses them towards different profiles of the models.

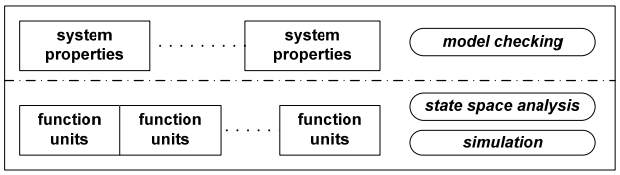


Fig. 11: The analysis framework of BitTorrent models.

As shown in figure 11, our analysis work is composed of two aspects. In the bottom, function unit is a basic functional flow of the protocol execution, for example, a leecher asks the peer list of the sharing the file, or a leecher downloads a piece from another peer. Based on the same models, different specific initial marking assignments can form different function units, that is, different initial marking will result in different executions of the protocol. Several function units could execute sequentially or concurrently to form a more complex functionality. We adopt simulation to validate the function unit, together with some state spaces analysis to check basic properties of them, such as boundedness, liveness, or deadlock checking. Just like the relationship between unit testing and system testing, based on the correct function unit, some higher properties of the protocol system should be verified, such as no out-of-orders executions, or a peer downloads pieces randomly, etc. Such properties are usually described as some temporal logics and verified using model checking technologies. In this paper, we use ASKCTL [18], provided by the CPN Tools, to describe such properties, and exemplify some key properties verification processes based on abstract models. From the point of view of model validation, function units simulation and analysis help validate the effectiveness of protocol detailed behaviors, and higher properties checking help verify the satisfiability to protocol requirements.

5.1 Function Units Validation

During the process of model construction, simulation is frequently performed to check whether the model behaves as expected. Because the simulation has immediate visual feedbacks, it is quite useful in finding modeling errors. Especially, the single-stepping through the simulation is very helpful to understand the details of original protocol specification, and make necessary refinement to CPN models. It is a good way to modify the model immediately when such simulation is performing.

According to sufficient simulations of BitTorrent CPN models, we find that most of concurrent behaviors existed in the model could be serialized. For example, when a piece packet is received, the BMSET should be updated and some new piece requests should be sent. These two behaviors execute independently, and if we model them as two independent substitution transitions, they will introduce concurrence in protocol running, together with large state space. In fact, those concurrent behaviors are not

intrinsically concurrent, and they could be serialized by assigning an execution order in the model manually. As discussed in section 4.6, we coercively arrange the execution order of these behaviors to effectively reduce much unnecessary state space.

Unfortunately, there are still some true concurrent behaviors happened in BitTorrent CPN models. For example, when a leecher receives a peer list from the tracker with at least two peer candidates, it will connect to both concurrently for different piece requests. As shown in figure 12, the left trace (nodes 11->12->15->19->24->30 ->37->45…) and the right trace (11->13->16->20->25->31->38->46…) respectively indicate that a leecher request pieces from two different peers, and the other traces all present the interleaving executions between these two behaviors. In fact, most of such traces are meaningless for analysis because they are too detailed. Some conflict access of significant data is worthy of consideration, and simulation related capabilities in CPN Tools are strong enough to validate such behaviors because of the visual feedbacks to check whether conflicts really happen. According to such observations and inferences, we generate several function units based on both functionality of protocol and true concurrent behaviors, that is, each function unit represents a relatively independent functional flow of protocol executions with no or controllable true concurrent behaviors. These function units should cover all paths of the model, and their sequentially or concurrently executions form all feasible functionalities of original specification. Towards each function unit, we perform both simulation and state spaces based static analysis to validate the reliable execution of such function unit.

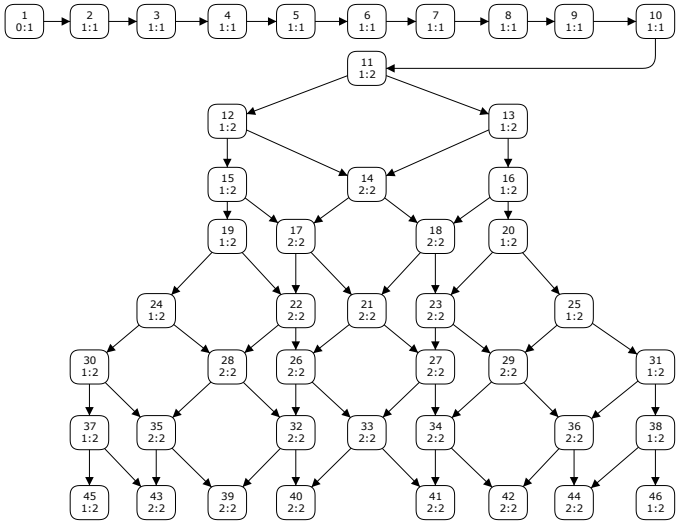


Fig. 12: The example state spaces of BitTorrent CPN models.

In our analysis, four function units are designed:

(1) *Leecher1* firstly asks *Tracker* for peer list of the sharing file (supposed to composed of two pieces), then *Tracker* replies with list containing *Leecher2* and *Seed*, where *Leecher2* has one piece and *Seed* has entire file.
(2) *Leecher1* connects to *Leecher2*, and download one piece without further pieces requests.

(3) *Leecher1* connects to *Seed*, downloads two pieces, and announces *Leecher2* that it has the entire file using piece having packet.
(4) *Leecher1* executes the rarest first piece selection algorithm, and *Seed* executes choking algorithm when receiving piece request from *Leecher1*.

Function unit (2) and (3) can execute concurrently after (1) has finished, and (4) can execute sequentially before (3). These four function units cover all paths of CPN models and major functionalities of BitTorrent systems according to the original specifications. After such function units division, it is simple but representative for model analysis. Firstly, we perform sufficient simulation towards these function units to remove unnecessary concurrence and refine models. The refinement process includes two aspects, correcting the inaccurate behaviors modeling and abstracting the redundant behaviors without meaningless details. Then, based on these function units, automatic state spaces analysis provided by CPN Tools is performed to check basic properties of such function units. The analysis results of above four function units are overviewed in table 1.

Table 1: State space analysis result of four function units.

function units	*state space status*	*nodes*	*arcs*	*boundness*	*home markings*	*dead markings*	*live transitions*
1	full	10	9	normal*	last marking+	last marking	none
2	full	100	99	normal	last marking	last marking	none
3	full	4910	8978	normal	none	last marking	none
4	full	18	17	normal	none	none	all
1+2+3	partial#	60358	111126	—	—	—	—

* *"normal" indicates no exceptions existing in boundness checking.*
\+ *"last marking" indicates the state in models where function unit executes successfully and terminates.*
\# *"partial" indicates it can not generate full state spaces under the time limitation of 1000 seconds.*

5.2 Model Checking of System Properties

After above validation procedures, function units are verified thoroughly. Each function unit execution starts from a specific initial marking and with no or controllable concurrent behaviors, therefore, the state space generated for this function unit only contains the states that could be reached from that initial marking, and the size of such state space is usually not too large to analysis. However, if we focus on checking higher system properties, such as mutual relationship among function units or system level requirements, we need full state space to enumerate every possible execution of protocol systems. Unfortunately, based on BitTorrent CPN models constructed in section 4, the state space explosion happens that we can not utilize advanced state spaces queries [17] or model checking technologies [18] to verify such properties.

In this paper, instead of considering the concrete full state space generated from original CPN models constructed in section 4, we check higher properties over a finite abstraction. According to modeling architecture mentioned in section 3, the abstract models only cover network, node and interaction layers. More specifically, the network and node layers in abstract models remain the same as the original models, and the interaction layers in abstract models are modeled as leaf page instances without

substitution transitions, that is, replacing the substitution transitions in original models with ordinary transitions. Besides, the abstract models contain some new places representing key data structures, the same as that appeared in original models. Such abstraction takes effect just because on one hand the functionalities of original transaction layer or algorithm layer model have been validated, the ordinary transitions could represent equal and valid functionalities as original substitution transitions, and on the other hand, original transaction layer models are always independent in functionalities with each other except for accessing the common data structures, so we reserve these data structures in new abstract models to keep the interaction relationship between corresponding behaviors. The abstract models could effectively relieves the notorious state space explosion, and make model checking feasible. This kind of abstraction could be considered as a kind of over-approximation. On checking higher properties on the abstract models, if the property passes verification, it also holds in original detailed models. Otherwise, simulation is utilized to find out the reason of failed verification: modeling error, protocol defects, or inaccurate abstraction.

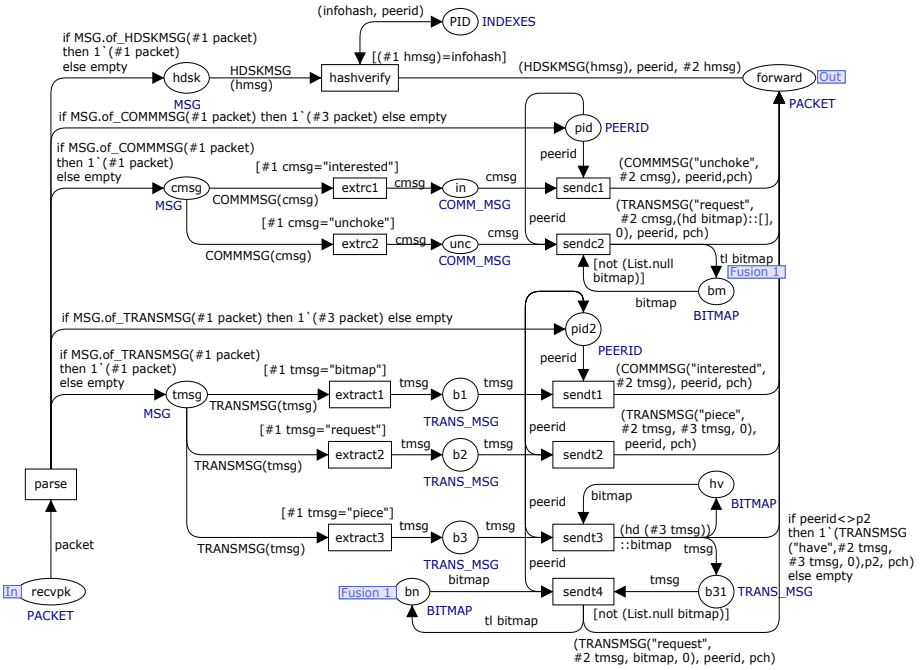


Fig. 13: The abstract receiving behavior model of a leecher.

As a representative, figure 13 presents packet receiving behaviors of a leecher in abstract models. Compared with figure 7, it has no substitution transitions, and only focuses on protocol communication behaviors. The newly constructed abstract model has 10 page instances in total. We assign different initial markings to conduct different execution flows. The rationality of such assignments has been validated in function unit analysis. As an example, we consider the concurrent execution of function units (2) and (3) defined in section 5.1. The full state space contains 9180 states and 22546 arcs. There are no home markings, no live transitions and 16 dead markings.

Using “State Space to Sim” tool in CPN Tools, we could clearly observe the color tokens in certain places, which proves that those dead markings exactly correspond to different concurrent execution results. Based on full state space, we also perform higher properties verification using ASKCTL based model checking. For example, considering a situation that a peer receives a piece without having received a unchoke message before, we specify *BTFormula* to check such situation never happens. Because ASKCTL has no definition of formula like “A→B”, we use its equivalent form as “OR(NOT A, B)” instead. Figure 14 presents the property description and checking result. This kind of properties usually referred as safety properties, and hard to simulate manually. Its successful verification (val it = true: bool) indicates that both abstract and detailed models behave accurately according to the property. Many other safety properties could be checked effectively and efficiently in the same way.

```
use (ogpath^"/ASKCTL/ASKCTLloader.sml")

fun IsUnchoke a =
(Bind.receives1'sendc2 (1, {peerid=p2, cmsg=("unchoke",1), bitmap=[1,2]}) = ArcToBE a);

fun IsRecvPiece a =
(Bind.receives1'sendt3 (1, {peerid=p2, tmsg=("piece", 1, [1], 0), bitmap=[]}) = ArcToBE a);

val BTFormula =
INV(OR(MODAL(AF("Unchoke", IsUnchoke)), NOT(MODAL(AF("ReceivePiece", IsRecvPiece)))));

eval_node BTFormula InitNode;
```

```
val IsUnchoke = fn : Arc -> bool
val IsRecvPiece = fn : Arc -> bool
val BTFormula =
  NOT
    (EXIST_UNTIL
      (TT,
       NOT
        (OR
          (MODAL (AF ("Unchoke",fn)),
           NOT (MODAL (AF ("ReceivePiece",fn)))))))) : A
val it = true : bool
```

Fig. 14: The ASKCTL based model checking of higher properties.

From above property verification process, it is clear that this abstraction guided checking method not only takes full advantage of sufficient validation to function units, but also makes higher properties checking practical and effective. According to our limited experience on network protocols modeling and analysis, this method is always regarded as a cost-efficient choice.

6 Conclusion and Future Research Issues

BitTorrent is one of the most popular protocols used in P2P applications providing fast file distribution and effective file sharing. It has complex communications and concurrent behaviors, which are major hurdles for formal functional modeling and validation. In this paper, towards such complex protocol system, a hierarchical modeling architecture is proposed to facilitate modeling system functionalities into several abstract layers, and expressing behavior details accurately and flexibly. Then, we utilize CPN as an effective actualization of above modeling architecture to construct BitTorrent CPN models, taking full advantage of the industrial-strength modeling capabilities of CPN. Several exemplified CPN pages are presented, and corresponding

modeling techniques are discussed at the same time. To the best of our knowledge, it is the first time to present a functional formal model of BitTorrent in peer level. It could not only be served as an unambiguous and visual formal specification for different system implementations, but also facilitate the behaviors simulation and properties verification of BitTorrent. At last, taking full advantage of strong analysis capabilities in CPN Tools, efficient and sufficient BitTorrent models validation is performed in both function unit level and system requirement level using simulation, state space analysis and model checking technologies together. They are used towards different abstract levels of above models to validate the effectiveness of models, and check whether these models satisfy the requirement properties of BitTorrent.

As for the future research, time factors will be introduces into current CPN models, because the choking algorithm and some peer selection algorithms are essentially time-driven. Besides, continuous improvements of making our BitTorrent CPN models more complete and more efficient need inevitably further studies.

Acknowledgments: This work was supported by the National Natural Science Foundation of China under Grant No. 60863015, the Key Program of Natural Science Foundation of Inner Mongolia of China under Grant No. 20080404ZD20, and the ChunHui Program of the Ministry of Education of China under Grand No. Z2007-1-01042.

References

1. Bram Cohen. Incentives Build Robustness in BitTorrent. In Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems, page 5, Jun. 2003
2. D. Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. In Proceedings of the ACM SIGCOMM'04, pages 367~378, Aug. 2004
3. M. Barbera, A. Lombardo, G. Schembra, etc. A Markov Model of a Freerider in a BitTorrent P2P Network. In Proceedings of the IEEE Globecom'05, pages 985~989, Nov. 2005
4. Lei Guo, Songqing Chen, Zhen Xiao, etc. A Performance Study of BitTorrent-like Peer-to-Peer Systems. IEEE Journal on Selected Areas in Communications, Vol. 25, No. 1:155~169, Jan. 2007
5. Amir H. Rasti and Reza Rejaie. Understanding Peer-level Performance in BitTorrent: A Measurement Study. In Proceedings of the 16th International Conference on Computer Communications and Networks (ICCCN 2007), pages 109~114, Aug. 2007
6. Vivek Rai, Swaminathan Sivasubramanian, Sandjai Bhulai, etc. A Multiphased Approach for Modeling and Analysis of the BitTorrent Protocol. In Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS 2007), page 10, Jun. 2007
7. K.N. Parvez, C. Williamson, Anirban Mahanti, etc. Analysis of BitTorrent-like Protocols for On-Demand Stored Media Streaming. In Proceedings of ACM SIGMETRICS'08, pages 301~312, Jun. 2008
8. Kurt Jensen. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Springer-Verlag, 2nd edition, Vol. 1~3, 1997.
9. Kurt Jensen, Lars Michael Kristensen and Lisa Wells. Coloured Petri Nets and CPN Tools for Modeling and Validation of Concurrent Systems. International Journal on Software Tools for Technology Transfer, Vol. 9, No. 3-4: 213-254, Springer Verlag, Jun. 2007
10. CPN Tools. Online: http://wiki.daimi.au.dk/cpntools/cpntools.wiki.

11. Jonathan Billington and Amar Kumar Gupta. Effectiveness of Coloured Petri nets for Modeling and Analyzing the Contract Net Protocol. In Proceedings of the 8th CPN Workshop, pages 49~64, Oct. 2007
12. Marko Bago, Nedjeljko Peric and Sinisa Marijan. Modeling Bus Communication Protocols Using Timed Colored Petri Nets - The Controller Area Network Example. In Proceedings of the 9th CPN Workshop, pages 103~122, Oct. 2008
13. Jing Liu, Xinming Ye, Jun Zhang, etc. Security Verification of 802.11i 4-way Handshake Protocol. In Proceedings of the IEEE International Conference on Communications (ICC 2008), pages 1642~1647, May. 2008
14. Panagiotis Katsaros. A Roadmap to Electronic Payment Transaction Guarantees and a Colored Petri Net Model Checking Approach. Information and Software Technology archive. Vol. 51, No. 2: 235-257, Feb. 2009
15. Yanlan Ding and Guiping Su. A Reduction method for Verification of Security Protocol through CPN. In Proceedings of IEEE International Conference on Networking, Sensing and Control (ICNSC 2008), pages 73~77, Apr. 2008
16. Jinan Yi-xin, Lin Chuang, Qu Yang. Research on Model-Checking Based on Petri Nets. Journal of Software, Vol. 15, No. 9:1265-1276, Sep. 2004 (in Chinese)
17. Kurt Jensen, Soren Christensen and Lars M. Kristensen. CPN Tools State Space Manual. Jan. 2006
18. Allan Cheng, Soren Christensen and Kjeld H. Mortensen. Model Checking Coloured Petri Nets Exploiting Strongly Connected Components. Technical Report of Computer Science Department, Aarhus University, Denmark, Mar. 1997
19. Timo Latvala. Model Checking LTL Properties of High-Level Petri Nets with Fairness Constraints. In Proceedings of the 22nd International Conference on Application and Theory of Petri Nets (ICATPN 2001), pages 242~262, Jun. 2001
20. Lisa Wells. Performance Analysis using CPN Tools. In Proceedings of the 1st International Conference on Performance evaluation Methodologies and Tools (ValueTools 2006), page 10, Oct. 2006