*Proceedings of the 7th International Conference on Applied Informatics*
*Eger, Hungary, January 28–31, 2007. Vol. 1. pp. 193–201.*

# Modeling P2P protocols by cellular automata[*]

**Gábor Balázsfalvi, János Sztrik**

University of Debrecen, Hungary
e-mail: {gbalazsfalvi,jsztrik}@inf.unideb.hu

**Abstract**

Recent peer to peer (P2P) protocols have been investigated and tested by discrete-event network simulators mainly. These simulators are usually rather detailed and they allow individual settings for each peer, because every peer is an individual entity. One of the drawbacks of this approach is the very low simulation speed. To reach the steady state and to obtain the system parameters are really difficult tasks.

In this paper we present another kind of simulation model for P2P networks based on cellular automata (CA). One of the benefits of our model is the large amount of theoretical results associated with the global behavior of CA. We show how standard distributed protocols can be modeled by the help of CA. The correctness of the protocols can also be proved by this technique, or – in some cases – counter-examples may be found.

*Keywords:* modeling, peer to peer network, cellular automaton

*MSC:* C.2.4 [Computer-Communication Networks]Distributed Systems

## 1. Introduction

Modeling protocols and algorithms used by distributed systems, focusing on peer to peer (P2P) systems has become an important topic nowadays. Correctness of network protocols is usually either mathematically proved or tested by simulations. Those that are only simulated are not necessarily trustworthy; this is unacceptable for some sort of applications, for example, where high level of security and/or reliability is needed. More and more distributed protocols have come to light due to their benefits against the regular client-server conception. The price is that they cannot be modeled by the conventional tools using, for example, queuing

[*]Research is partially supported by Hungarian Scientific Research Fund-OTKA K 60698/2006 and OTKA T049409. This work was also supported by grants of Xerox Foundation UAC (No. 1478-2004), USA.

systems. They are usually designed on the desk and only simulation data are available at best. Our proposal gives more freedom in modeling distributed algorithms, and on the other hand it allows showing their correctness in a more appropriate way than traditional tools.

Our proposal is the usage of a rather old tool from a different area of computer science, i.e. the cellular automata (CA). This paper continues with a discussion of P2P networks and the algorithms we would like to model. Then we summarize the recent use of tools to model and analyze these protocols. After it we define precisely our CA concept, and we finish the paper by some examples and conclusion.

## 2. P2P Networks

### 2.1. Concept of P2P Networks

Distributed networks have existed since the 80s or before. The term P2P network became used with the concept of distributed file sharing. Early attempts were, for example, the Gnutella network and the Napster system. Up to that time, sharing data happened in the following way. The client wanting to share something had to look for a server with large amount of storage space and high bandwidth. Then this client uploaded the data there, and other people who needed the same data had to download from that server. If the data was popular, the server became very overloaded. On the other hand, if the shared content was useless, it just reserved costly space and network resources. Nevertheless, why is it necessary to store the data on a server, if it is already available from the original client? Actually it is not the case. Clients wanting to download the data can download from the original owner, and what is more, they can forward it to other clients. So the data travels from one peer to another, so P2P means the type of the layout. The peers become clients and servers at the same time. In general, every peer is equivalent to one another. Another important feature of this layout is that the peers are interconnected in an ad-hoc manner, because no peer is special. In general, no client knows in advance where the data travels from, and he is not even interested in these pieces of information, but knows the content only.

### 2.2. Use of P2P Networks

To solve the basic problems, these networks use well-known distributed algorithms, for example, for security or communication, but need also new, special algorithms. The above mentioned data sharing is just one example. It covers now the BitTorrent, the Gnutella, the Kazaa, and so on.

Another kind of use is the modern chat-voice-video clients, like Skype, Microsoft Live (MSN) Messenger, Yahoo Messenger, or the modern VoIP computer to telephone softwares, where usually clients sign up for a well-known server when they are online, and talk directly to each other when it is necessary.

### 2.3. Protocols to Model

In this part we would like simply to enumerate some protocols which we are interested in without attempting to be comprehensive. Probably the most important protocol is the delivery of a message. Here one of the interconnected peers creates a message it wants to send to the other peers or to a special client.

Solving the problem of message delivery, we can build up complicated protocols. An important one is the distributed synchronization. Here we use the messages as signals that spread over the peers and change their states. We will discuss about this protocol in details later in the section of examples.

Hash table protocols are even more complicated and they are building bricks of other protocols. Here the task is to store key-value pairs in a distributed manner. The pairs should be distributed uniformly among the participants. One from outside of the system has to know only one peer's address and the protocol to invoke the hash table, for example, as a web service.

On the top of distributed hash tables - and also without them - we can build different distributed content sharing algorithms. Some of them are widely used, such as the BitTorrent. The standard BitTorrent does not use distributed hash tables, but a central server to store metadata about the files that are stored in the system. On the other hand there are investigations to extend this protocol to be totally distributed. This is very useful when no central address can be assumed. For instance, we could talk about ad-hoc networks where the peers connect and disconnect in every minute, and there is not any standard one among them. In an ad-hoc network it is hardly possible to use any server based algorithm.

Last but not least we also mention the replication protocols. They are used by highly loaded content servers, for example, enterprise file servers that have to work in real time. The point is that the occasionally required data should be stored only once while popular data would be put on many servers. Those copies have only one common address. When the user goes to this address the system decides which server will serve him, and tries to distribute the load by redirections.

## 3. Network Modeling Technics

### 3.1. Stochastic Modeling

Traditional client server architectures and protocols over them are very often modeled by queuing systems. The server has usually one or more queues what the clients line up in. We can also associate different serving disciplines with these servers, for example, the clients can have priority, can be served parallel or one after the other; if so, we can choose between "last in first out" and "first in first out" queues. Also, queues can have finite or infinite capacity. The arrival of the clients and the serving time are independent random variables. All these things are mathematically clear. Easy setups can be solved on paper by hand, while more complicated ones are aided by computer tools.

The downside is that this model does not suit P2P systems. Peers are interconnected in an ad-hoc manner, but we cannot build connections or interactions into the model. There are some papers in the literature trying to model e.g. BitTorrent, but they usually go too far in generalization.

### 3.2. Discrete Event Simulation

Complex network systems are very often modeled by discrete event simulators in the last resort. They are disliked because the given results are not always reliable. Using such a simulator needs a lot of experience, and even in that case making mistakes is very easy. The simulation time is often very long, sometimes it takes days to get results with one setup of parameters.

To have some good words about them we can say that they are capable to model every sort of networks and protocols. If no known tool satisfies our needs, we can take some general simulator kernel and write our model programmatically in a few days. Then we can measure any parameter and compute any result that we need without elaborate mathematics or formulas. The quasi randomness can be ensured by semi random number generators, which also means that we can repeat the same sequence of events many times using different parameter setup in the system.

## 4. Cellular Automata

### 4.1. Definition

In this section we are going to define and describe exactly what CA are. First we have to define cellular space.

**Definition 4.1** (Cellular space). The cellular space (CS) is the $\mathbb{Z}^n$ n dimensional space together with one copy of a special deterministic finite automaton (DFA) in each cell. These copies are also referred to as cells. This automaton consists of a state set $S$, an initial state $s_0 \in S$, and a transition function $\delta : S^k \mapsto S$. We also have a $k$ dimensional vector $v$ of coordinates of $n$ dimension. This determines the so-called neighbors of the cell. Usually $v_0 = (0, 0, \dots, 0)$, but not necessarily. If the coordinates of the actual cell are denoted by $C$, the $\delta$ function must be applied to the states of the cells $C + v_0, C + v_1, \dots, C + v_{k-1}$ to get the new state of the actual cell. All of the mentioned properties of the copies must be common but the initial state.

The automata are working parallel and synchronized. They make one step at each time segment. The transition function collects the actual states of the neighbors - which is the initial one at the beginning - and makes the decision on the next state.

**Definition 4.2** (Cellular automaton). Let us be given a CS and the participating automata have a special state $s_f$ such that

$$\delta(s_f, s_{i_0}, , s_{i_1}, \dots, s_{i_{k-2}}) = s_f$$

for every state sequence $s_{i_0},, s_{i_1}, \ldots, s_{i_{k-2}}$, and we require the first parameter of the transition function $\delta$ to refer to the state of the actual cell. It is also important that this special state is not reachable from any parameterization of $\delta$. There are only finite number of cells having an initial state different from $s_f$. Then we call this space a cellular automaton (CA).

As an example, the two dimensional case is illustrated in Figure 1. Here we can see a rectangular "fence". All the cells inside this fence denoted by colors and pure white, are not in the state $s_f$, but the cells outside of the "fence" and the "fence" itself is in this special state. It is clear now that the cells outside of the "fence" will remain in the $s_f$ state forever, while the ones inside will never change into this state.

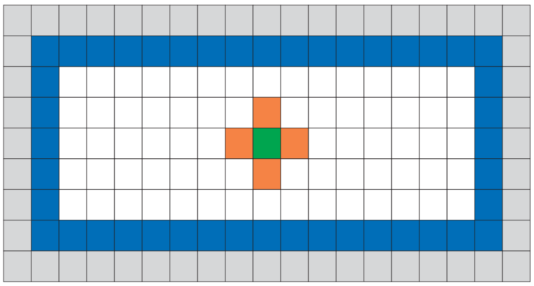
Figure 1: 2 dimensional cellular automaton

## 4.2. Neighborhoods

The $\delta$ transition function is special in a sense that it does not depend only on the cell's own state, but also on some other cells' state. This was defined in the previous part. Now we show some special neighborhoods often used in the literature.

All used neighborhoods include the actual cell itself. The Moore neighborhood includes the cells the coordinates of which differ from the actual cell in any place but with at most one. In one dimension these are the left and the right neighbors, while in two dimension, if the actual cell has coordinates $(0, 0)$, it contains $(-1, -1), (0, -1), (1, -1), (-1, 0), (0, 0), (1, 0), (-1, 1), (0, 1), (1, 1)$. It has also a modified version where only the nonnegative coordinates play role. The other important neighborhood is named after Von Neumann. This includes the cells with coordinates not differing more than one in sum from the actual cell's coordinates. In one dimension this means the same as before, but in two dimensions, it contains only 5 cells: $(0, -1), (-1, 0), (0, 0), (1, 0), (0, 1)$. We show these examples in Figure 2. We can certainly consider also different constant than one.

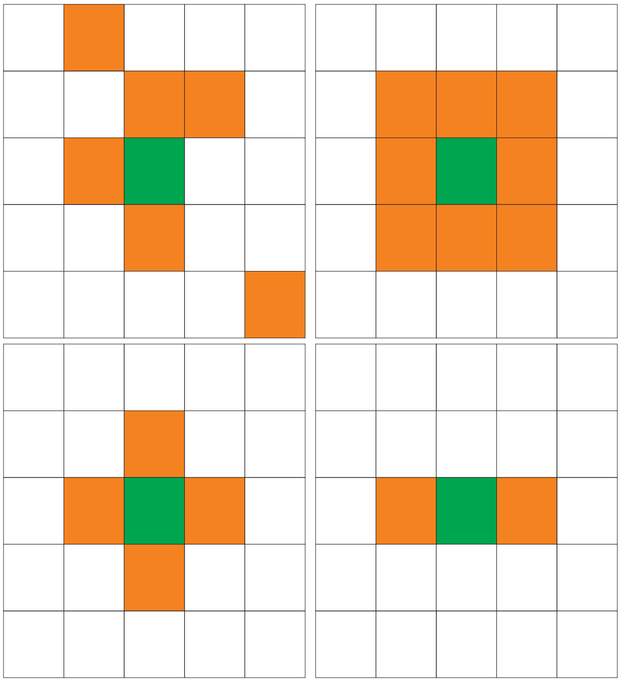

Figure 2: Neighborhoods. From left to right and up to down: 2 dimensional arbitrary, Moore, Von Neumann, and either Moore or Von Neumann in 1 dimension

## 5. Examples

### 5.1. Message Delivery

This example uses any number of peers interconnected according to the one dimensional Von Neumann neighborhood. The first peer wants to send a message to all the others. For this, we use a special state in the state set $S$ to denote the message. The transition function we use changes into that message state, when sees it at one of its neighbors and has never seen it before, i.e. the actual cell is still in the initial state. It can also remember where the message came from (left or right). When it sees the message at its own cell in the next step it erases the message, and remembers that it has already met this message. By this very simple algorithm, the message travels through all the peers, reaching them only once. By using more states, the "speed" of the message can be decreased to moving one in every two steps ($\frac{1}{2}$), two in every five steps ($\frac{2}{5}$), or to any rational number between 0 and 1.

## 5.2. Synchronization

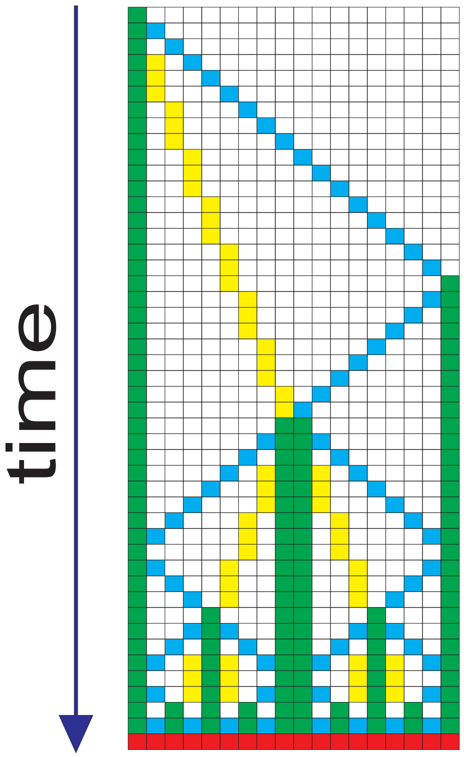

Figure 3: Synchronization process

The next example is the synchronization problem. We have some peers connected to each one, again according to the one dimensional Von Neumann neighborhood. Their task is to get into a common state exactly at the same time without even knowing how many they are. For this, the first one, the leftmost, sends a message towards the last one, the rightmost, with a speed of one, and another with a speed of $\frac{1}{3}$. After this, the first peer changes its state to a “ready to fire” state. When the first message reaches the rightmost peer, it sends back the same message. When this message meets the other one traveling slower, the actual cell is starting to behave like the first one: it sends two messages to the right; and additionally it sends the same kind of messages to the left, and changes its state to a “ready to fire” state. Sooner or later every peer will be ready to fire due to the recursion. When a cell becomes ready, and all its neighbors are ready too, it can fire. You can see this all in Figure 3.

### 5.3. BitTorrent Simulation

Our last example is the model of the BitTorrent file sharing system. In BitTorrent, the data is sliced to hundreds of pieces. Clients download from one other these small junk of data parallel. However, here we use only two pieces of data to distribute among the participants for the sake of simplicity. There are 9 peers and they are interconnected according to the two dimensional Von Neumann neighborhood. At the beginning, the middle one has all the two pieces while the other 8 do not have anything. Then as time goes on, the neighbors copy the pieces one after the other. First they try to download from the central peer, but when another neighbor has it already, they can take the piece from that one too. At the end, the 4 peers in the corner have all the pieces too, even though they do not flank on the central peer. A snapshot of this process can be seen in Figure 4.

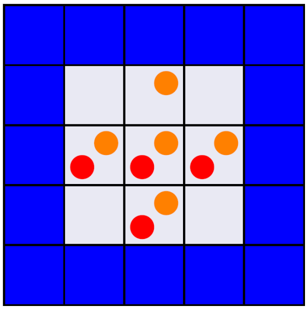

Figure 4: BitTorrent simulation

## 6. Conclusion and further work

In this paper we pointed out the drawback and the benefits of the standard modeling methods used to model P2P networks. Then we proposed a new method for this task. This new method uses cellular automata, which are also defined in this paper. We showed the way of modeling some often used protocols by cellular automata too. We plan to create a modeling tool which works with cellular automata. We also would like to model some other protocols, e.g. distributed hash tables, and obtain the properties of that protocols by the modeling tool.

## References

[1] Lindenmayer, A., Rozenberg, G., Automata, Languages, Development, *North-Holland,* (1976).

[2] Farmer, J. D., Toffoli, T., Wolfram, S., Cellular Automata, *Proceedings of an Interdisciplinary Workshop at Los Alamos,* New Mexico, March 7-11, (1983), North-Holland, (1984).

[3] Mullender, S., Distributed systems, *Wokingham: Addison-Wesley,* 2nd ed., (1993).

[4] Anceaume, E., Puaut, I.: Performance evaluation of clock synchronization algorithms, *Technical report,* RR-3526, INRIA, Rennes, France, (1998).

[5] Tanembaum, A., van Steen, M., Distributed systems: Principles and paradigms, *Prentice-Hall*, (2004).