A Mathematical Analysis of Prophet Dynamic Address Allocation

Cédric Lauradoux, Marine Minier

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Mathematical Analysis of Prophet Dynamic Address Allocation

Cédric Lauradoux — Marine Minier

N° 7085

October 2009

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--7085--FR+ENG

A Mathematical Analysis of Prophet Dynamic Address Allocation

Cédric Lauradoux , Marine Minier

Thème COM — Systèmes communicants
Projet SWING

Rapport de recherche n° 7085 — October 2009 — 15 pages

Abstract: Prophet is a dynamic address allocation protocol described at INFOCOM 2003. This protocol is based upon a family of pseudo-random generators. The goal of Prophet is to establish an addresses scheme free of conflict. The addressing capabilities of Prophet depend on the underlying properties of the pseudo-random generators. The different pseudo-random generators proposed in Prophet are analyzed and the limits of the scheme are exhibited. Most notably, the periods of the generators limit the addressing capabilities of a node and the fact that Prophet is collision-free. In this research report, we show that the underlying assumptions made in Prophet can not be met by pseudo-random generators.

Key-words: Dynamic addresses allocation, pseudo-random generator

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

A Mathematical Analysis of Prophet Dynamic Address Allocation

Résumé : Prophet est un protocole d'allocation dynamique d'addresse présenté à INFOCOM 2003. Prophet permet de générer des adresses libres de collision sans nécéssité de communications additionnelles que celle liées à l'allocation propre des adresses. Ce protocole s'appuie sur deux hypothèses importantes: l'existence de générateurs pseudo-aléatoires de grande période et la faible probabilité d'obtenir deux fois la même valeur dans deux séquences pseudo-aléatoires. Les capacités d'adressage de Prophet dépendent des propriétés sous-jacentes des générateurs employés. Pour implémenter leur protocole, les auteurs de Prophet ont proposés l'emploi de générateurs pseudo-aléatoires de type linéaire congruentiel. Les schémas proposés par les auteurs sont analysés. Il en résulte que Prophet n'est pas capable de vérifier l'hypothèse sur la période des séquences générées différentes. Ceci implique que Prophet n'est pas capable de supporter toutes les configurations possibles d'un réseau dynamique sans sacrifier l'unicité des adresses générées. Notre conclusion est que les hypothèses utilisées dans Prophet ne sont pas compatibles avec l'emploi des générateurs pseudo-aléatoires.

Mots-clés : Allocation dynamique d'adresses, générateurs pseudo-aléatoires

1 Introduction

Dynamic address allocation is a fundamental problem in mobile ad-hoc networks (MANETs). Address allocation preceeds any further operations in networking. This task is particularly difficult when there is no access to a centralized infrastructure, like in MANETs. The challenge consists in providing **unique addresses** to the network nodes using parallel and independent processes. Several propositions have been made to solve this problem. A complete survey on this problem can be found in [BCM09]. Four classes of algorithms have been identified so far: conflict-detection allocation [PRD00], conflict-free allocation [MDMD01], best-effort algorithms [NP02] and to conclude the Prophet algorithm [ZNM03]. Prophet is a particularly intriguing algorithm because it appears as a significant breakthrough in mobile networking. It provides an addresses allocation scheme free of conflict with a low complexity, a low communication overhead and a low latency.

The Prophet protocol is investigated in depth in this paper. Prophet is based on assumptions related to random number generation and the authors made an intensive used of linear congruential generators (LCGs). Then, we show that the underlying idea of Prophet is similar to the problem of parallel pseudo-random generators [Dur89]. This problem comes from the parallel computing and it is more specifically related to distributed Monte Carlo simulation [Nie92]. Indeed, one of the fundamental assumption of Prophet consists in using sequences of large period as in parallel computing. Moreover, the authors of Prophet have indeed used two well-known strategies to transform an LCG into a parallel random-number generator. Unfortunately, there exists two fundamental differences between the goal of Prophet and the problem discussed in parallel computing. First, the parameters used in the LCGs for Prophet are chosen dynamically and randomly while they are chosen statically in parallel computing. This fundamental difference allows to have some control on the properties of the random numbers produced in parallel computing. In the case of Prophet, this is clearly not the case. Second, Prophet assumes that the parallel sequences are collision free. This strong assumption not found in parallel computing can not be unfortunately satisfied using parallel pseudo-random generators. Therefore, we show that Prophet can not achieve its main goal: unique addresses allocation (see Figure 1).

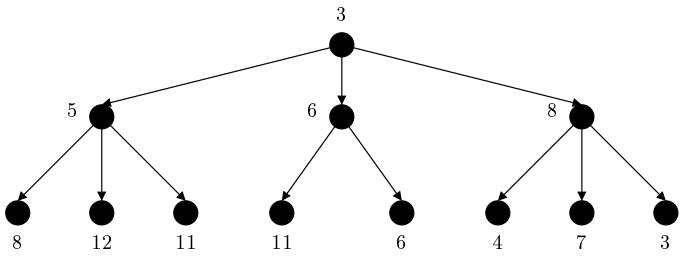

Figure 1: An example of addresses assigned with Prophet for $\mathcal{N} = 12$.

A definition of dynamic address allocation is given in Section 2. The principles and the assumptions used in the design of Prophet are described in Section 3. Then, we find particularly useful to analyzed the "toy example" used by the authors to present Prophet (Section 4). The key definitions of pseudo-random generators are remainded in this section. The practical pseudo-random generators proposed for Prophet and the possibility to achieve collision-free dynamic address allocation are discussed in Section 5. Finally, we conclude the paper.

2 Definitions

The goal of this section is to establish clearly the goal of a dynamic address allocation algorithm. The definition of such an algorithm and of an address graph are given. The address graph describes the address allocation scheme. A dynamic address allocation algorithm must be able to provide addresses for any possible address graph. Those definitions are very important to understand the limitations of Prophet.

Definition 1 *An **address graph** is an acyclic oriented graphs of arbitrary indegree and outdegree.*

The edges of the address graph represents the addresses dependency. If there is an edge from A to B, it means that A has assigned an address to B. There is at least a network connection between two nodes connected by an edge at the time of the address allocation. But two disconnected nodes of the address graph can have a network connection. The assignment scheme for the addresses may be related to another network mechanism.

The *indegree* d_i of a node corresponds to the number of addresses assigned to this particular node. A given node can have an arbitrary number of addresses. The *outdegree* d_o is the number of times that a given node has assigned addresses. For simplicity and without loss of generality, only address graphs for which the indegree of any node is $d_i \leq 1$ are considered in the following.

Definition 2 *A **dynamic addresses allocation algorithm** consists in giving a unique label to the $\mathcal{N}$ nodes of any possible address graphs.*

The representation of the address assignment as an acyclic oriented graph is very useful to capture the complexity of dynamic addresses allocation and the limitations of the current approaches to solve the problem. In a network, several nodes can start to assign addresses to other nodes. These particular nodes are called root and they have an indegree $d_i = 0$. The two extreme setups for a dynamic network are:

- An address graph with a node A(0,$\mathcal{N}-1$) (Figure 2 (a)). $\mathcal{N}-1$ nodes request an address to a single root. This case corresponds to the classical address allocation with a centralized infrastructure.
- An address graph in which all the nodes have an indegree and outdegree equal to 0 (Figure 2 (d)). Each node chooses for itself a given address, *i.e.* each node is a root.

Between those two extreme setups, any configuration is possible. For instance in wireless sensors networks (WSNs), a single root can assign the addresses of all the nodes which are distant from one hop (see Figure 2 (a) and (c)). In an *ad-hoc networks*, it is possible to have k roots in order to reduce the delay to obtain an address. In this particular case, the overall graph can be viewed as a single network or as k networks (sub-graph) with one network associated to each root. If the graph is considered as a single network, it implies that all the sub-networks are merged. Therefore, our definition is enough to cover operations like merging. In their paper [ZNM03], the authors provide materials for address graphs with a single root.

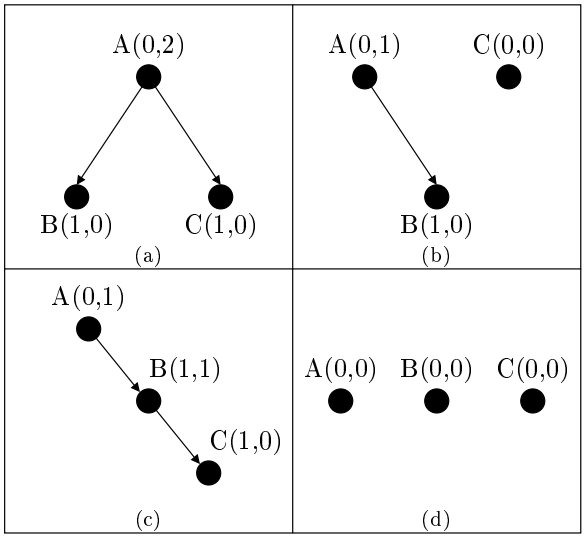

Figure 2: Different setups for an acyclic oriented graphs of indegree $d_i \leq 1$ with $\mathcal{N} = 3$, A(i,j) with $d_i = i$ and $d_o = j$.

3 Prophet: principle and hypothesis

3.1 Architecture of the scheme

In Prophet, each node received a black box (see Figure 3) allowing him to generate new addresses upon request. This black box is initialized with the node address a and a seed x_0. The seed is used to initialize an internal state x_t which is updated by the function g. This function g can be a counter (incrementation) as any other function. The internal state x_t and the address a are used to produce new addresses using the address derivation function f. It should be noted that the authors in their paper [ZNM03] do not mention explicitly the existence of an update function. However, it simplifies greatly our analysis.

Using this representation, one may find the problem considered by Zhou et al. very similar to the problem of parallel random generators [Dur89, Hal89]. Distributed computations like simulations make an intensive use of random numbers. Each threads of the simulation may need to have its own stream of random numbers such that all the streams of random numbers are uncorrelated. However, this problem is essentially static (everything is known in advance: number of computers, topology).

A first limitation of Prophet is the lack of initialization algorithm. Prophet can not be used when each node chooses for itself an address (all the nodes have an indegree and outdegree equal to 0). Prophet anwers the issue of address derivation and it is not suitable

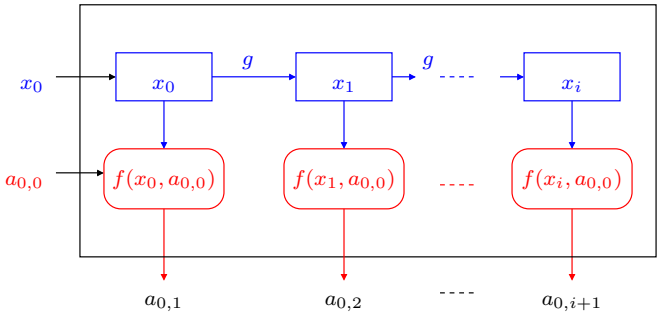

Figure 3: A view of Prophet.

when several roots assign addresses. It can not be used for merging networks contrary to the authors claim. In the following, a single node with $d_i = 0$ is assumed.

3.2 Update function and address derivation function

The main assumptions on the update function and on the address derivation function are given here as in the original paper:

Assumption 1 *The interval between two occurrences of the same number in a sequence is extremely long.*

Assumption 2 *The probability of more than one occurrence of the same number in a limited number of different sequences initiated by different seeds during some intervals is extremely low.*

The Assumption 1 corresponds with the requirement of pseudo-random generators: the period T of a sequence S must be as large as possible. The readers may find helpful informations on pseudo-random numbers in classical textbooks [Nie92, Knu97].

The Assumption 2 is a property that must be satisfied by comparing several sequences. It means that the repetition of a given value in several sequences must be really improbable.

The remaining parts of the paper concern the design of f and g such that Assumption 1 and 2 are verified. First, the toy example given in Section III.B of the original paper [ZNM03] is dissected. Then, we deal with the practical design of f and g proposed by the authors.

4 Prophet: the toy example

To illustrate the basic idea of their paper, Zhou et al. [ZNM03] propose to use the folowing functions:

$$\begin{aligned} f(a, x_t) &= a \times x_t \times 11 \bmod 7 \\ x_{t+1} &= g(x_t) \\ &= f(a, x_t) \end{aligned} \tag{1}$$

with a a given address. When a new address is provided, a new seed is computed. In this example, the new seed is the address itself. Here, the update function g and the address derivation function are the same. For the address graph depicted in Figure 4 (a), Prophet allocation with Equation 1 provides the addresses shown in Figure 4 (b). The addresses are generated by two pseudo-random generators (Figure 4 (c)).

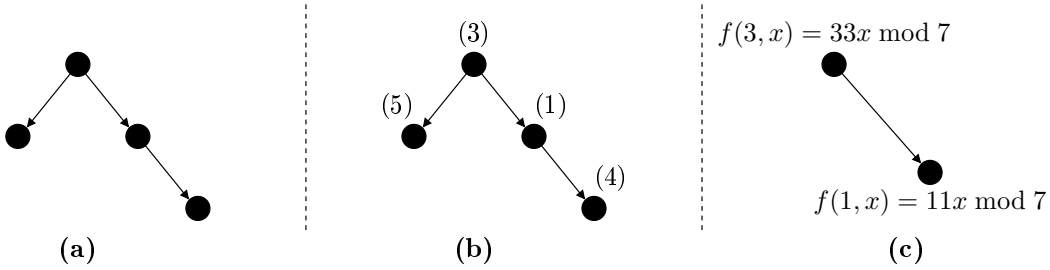

Figure 4: Example of allocation with an initial address $a = 3$.

It must be noticed that the multiplication by eleven is a simple permutation of the address a. f can be written:

$$\begin{aligned} f(\gamma, x_t) &= \gamma \times x_t \bmod 7 \\ \gamma &= a \times 11 \bmod 7 \end{aligned}$$

In this later form $f(\gamma, x_t)$, those functions are clearly identified as the multiplication linear congruential generators (MLCGs). This is a special case of the linear congruential generators defined by Lehmer [Leh51].

4.1 Linear Congruential Generators

A linear congruential generator of multiplier b, modulus m and increment c is defined by:

$$x_{t+1} = b \times x_t + c \bmod m. \tag{2}$$

This family of generators has been well studied and numerous works exist on LCGs [PM88]. Most notably, they have statistical weaknesses which made them unsecure for cryptographic applications [Knu85, Ste87]. Some basic properties of MLCGs are now remained.

For the MLCGs ($c = 0$), it is well known that the period T, *i.e.* the smallest integer such that $x_{t+T} = x_t$, is the smallest integer k such that:

$$b^k \equiv 1 \bmod m.$$

If a and m are coprime, we state that using the Euler-Fermat theorem:

$$b^{\phi(m)} \equiv 1 \bmod m \tag{3}$$

where $\phi(m)$ is the Euler's totient function, the number of positive integers less than or equal to m that are coprime to m. Therefore, the period T can not exceed $\phi(m)$ which is maximal if m is a prime, *i.e.* $\phi(m) = m - 1$. An integer b verifying Equation 3 is called a primitive root modulo m. There is $\phi(m-1)$ primitive roots modulo m.

4.2 Analysis

In this example, each node is assigned at the same time with an address, a seed and a particular instance of a MLCG with modulus m. This technique is known as the "different multipliers" strategy in parallel computing [Dur89]. The period T represents the number of addresses that can be allocated by a node. If the period T is not large enough, the Assumption 1 can not be met.

This strategy failed in the context of Prophet because the multipliers are chosen randomly over $[1, m-1]$. The probability to have $T = m - 1$ is equal to:

$$P[T = m-1] = \frac{\phi(m-1)}{m-1}.$$

In the given example ($m = 7$), this probability is only $\frac{1}{3}$ and more generally for $m - 1 > 6$:

$$\frac{\sqrt{m-1}}{m-1} \leq P[T = m-1] \leq 1 - \frac{\sqrt{m-1}}{m-1}.$$

Assuming that we have successfully allocated $i - 1$ addresses, *i.e* the addresses are unique and the period associated to an address is always maximal, the probability for the i-th allocation to succeed is:

$$P[T_i = m-1] = \frac{\phi(m-1) - i}{m-1}$$

with T_i the period associated to the i-th address. This result strongly limits the addressing capability of the scheme. The more addresses are allocated with such a scheme, the higher is the probability that we get a degenerated function f. Prophet can provide conflict-free addresses for all topologies if a node has an address that implies a low period pseudo-random generator

A second problem arises with fixed points. If the Equation 1 is used with $m = 7$, it is easily seen that the initial value 2 always produces 2 as an address. This problem can be solved if a node which received an address searches and prevents the use of a fixed point.

A third problem arises when the Assumption 2 is considered. Let consider the address graph described in Figure 5 (a). In this case, the pseudo-random generators used to assign the addresses (Figure 5 (c)) have full period (see Figures 6 and 7). However, we still have a collision on the address assigned (Figure 5 (b)).

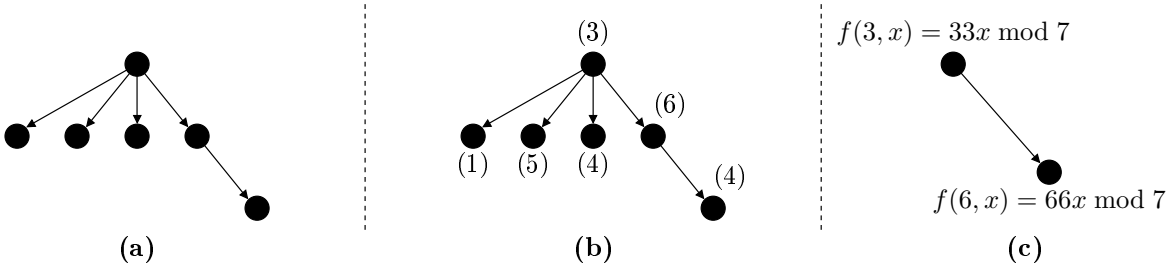

Figure 5: Example of allocation with a collision.

In fact, the Assumption 2 can be written:
$\nexists i < \sigma$, with $\forall x, y \in [1, m-1], \forall x', z \in [1, m-1]$, $x \neq x'$ and $z \neq y^i \bmod m$ such that:

$$x \times y^i \equiv x' \times z \bmod m$$

where σ is a fixed threshold. The authors do not see any other way to verify this property than exhaustive search ($\mathcal{O}(m^4)$). This property is generally not verified by parallel pseudo-random generators.

To conclude this analysis, the use of MLCGs for the implementation of Prophet is not recommended because the periods of the different generators are difficult to control (Assumption 1) and because no expectation on the probability of collisions between the different streams can be found (Assumption 2).

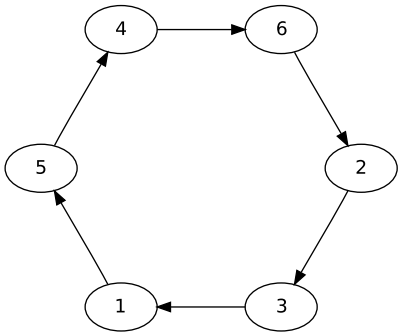

Figure 6: Behaviour of the generator for $a = 3$.

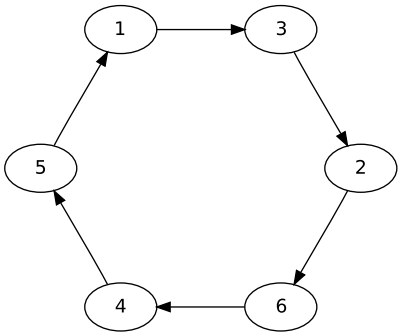

Figure 7: Behaviour of the generator for $a = 6$.

5 Analysis of the practical scheme

5.1 Description and Analysis

The authors proposed an other scheme to implement Prophet. In their practical construction, the update function g and the address derivation function f are different:

$$\begin{aligned} f_a(x_t) &= [a + h(x_t) \bmod m] + 1 \\ h(x_t) &= \prod_{i=1}^{n} p_i^{x_t^i} \\ x_{t+1} &= g_j(x_t) \\ &= (x_t^1, x_t^2, \cdots, x_t^j, \cdots, x_t^n). \end{aligned} \tag{4}$$

The function h is the product of the n first prime numbers p_i put to the power x_t^i. The internal state x_t is a vector of n integers $x_t^1, x_t^2, \cdots x_t^n$. This vector can be viewed as a virtual coordinate system in the address graph. The update function g_j consists in increasing by one the coordinate x_t^j. The value j is fixed for a given node. The modulus m is the bound on the number of addresses which can be allocated. When an address is requested, the internal state is first updated and then, the function f_i is applied. Once a new address $b = f_a(x_t)$ is produced by a node a the current state of the node is sent as a seed for the new node and the new value $j_b = j_a + 1$. The root node has for address a and its internal state is $\forall i \in [1, n],\ x_0^i = 0$. An example for $n = 4$ and with $x_0 = (0 \parallel 0 \parallel 0 \parallel 0)$ is given in Figure 8.

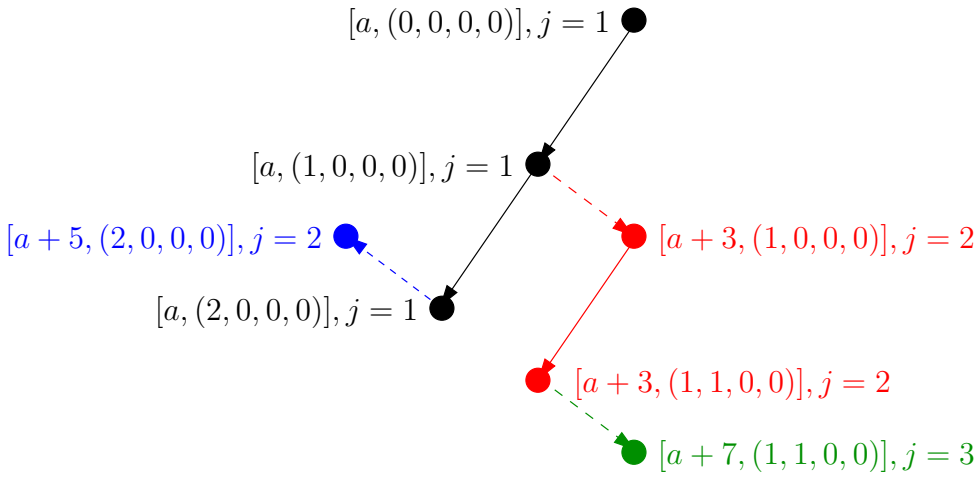

Figure 8: Example of allocation with an initial address a.

j	Function	x_{t+j}	Period T
1	$f_a(x_{t+j}) = (a + 2^j \bmod m) + 1$	$(j \parallel 0 \parallel 0 \parallel 0)$	16
2	$f_a(x_{t+j}) = (a + 2 \times 3^j \bmod m) + 1$	$(1 \parallel j \parallel 0 \parallel 0)$	51
3	$f_a(x_{t+j}) = (a + 6 \times 5^j \bmod m) + 1$	$(1 \parallel 1 \parallel j \parallel 0)$	76

Table 1: Period of several functions for $m = 151$.

Despite its apparent complexity, this period of the sequence generated is relatively easy to obtain. The key derivation function f_a can be written for $j = 1$:

$$f_a(x_{t+k}) = [a + 2^k \bmod m] + 1.$$

The period T is given by $T = k - i$ where i is fixed integer and k is the smallest integer such that:

$$2^k \equiv 2^i \bmod m \text{ and } k > i. \tag{5}$$

This result can be easily extend to other values of j and different internal states x_t. For instance we have computed the period of several functions for $m = 151$.

The previous table clearly show that we obtain as in Section 4 a family of generators with a period difficult to control. The Assumption 1 can not be met by this generator.

As shown previously, the period T of the generator is exactly the same than in the previous. In fact, in this last case, we are looking for powers of prime numbers that have a maximal period, i.e. that are primitive root mod m. In the same way, the number of primitive root mod m is $\phi(m-1)$ and strongly depends on the decomposition of $m - 1$.

In the Prophet practical scheme, with m a prime number, the four numbers 2,3,5,7 must have a maximal order. Due to the probabilities computed in Section 4, this is hard to reach. With the prime number $m = 1031$, $\phi(m)$ is equal to 408, the order of 2,3 and 5 is 515 and the order of 7 is 206. So, finding an m value for which the 4 first prime integers have a maximal order has a really low probability. And the general probability to obtain a primitive root is exactly the same than the one given in Section 4.

5.2 Parameters choices and examples

The implementation of this scheme require to define three parameters, (1) the modulus m, (2) the first address a, and (3) n the number of prime numbers used and the internal state. In the original paper [ZNM03], there is no discussion about this three parameters and no practical parameters are provided. Later in [Zho08], Zhou provide details on the choice of m and n.

Choice for m. The modulus m must be chosen carefully regarding the size of the network $\mathcal{N}$ and the previous remarks on the period. The original paper, it is considered that the modulus m should be the address range +1.. In [Zho08], it is considered that m must be the highest prime number less or equal that the address range. However, it is straightforward

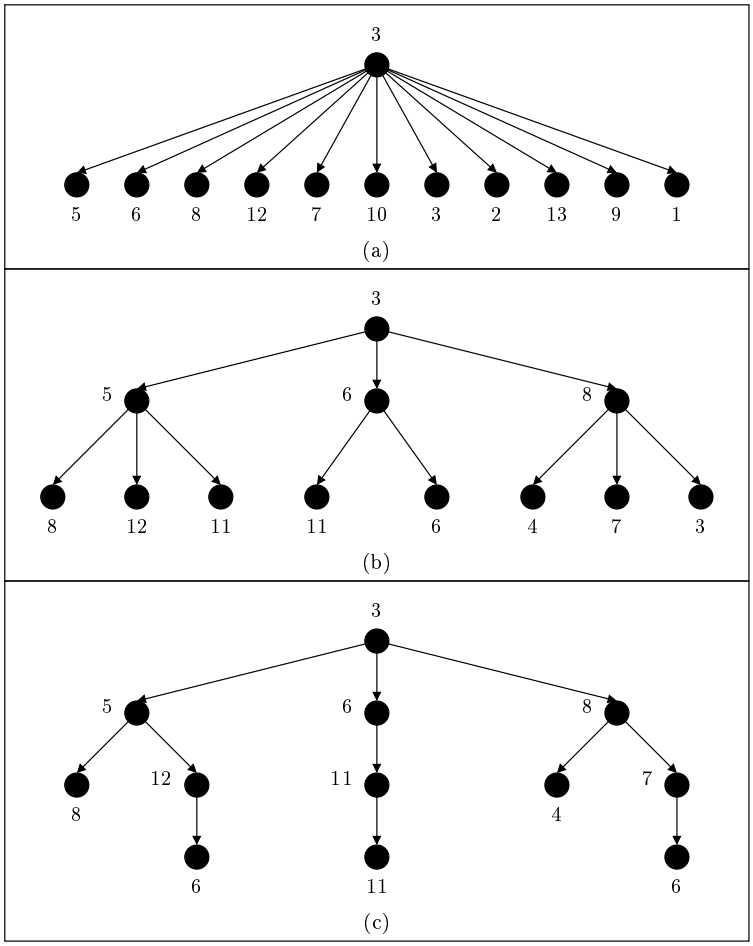

Figure 9: Address assignment obtained with Prophet for $\mathcal{N} = 12$.

that this solution is ill-fated when the address range match the network size $\mathcal{N}$: it is not possible to assign $\mathcal{N}$ unique addresses from computations in the range $[0, m-1]$ if $m < \mathcal{N}$. Clearly, a better choice would have been to consider that m is the smallest prime number greater or equal $\mathcal{N}$.

Choice for a. This parameter can affect the probability of collisions since the value a can appear later in another pseudo-random sequence. We assume that this value is chosen randomly in $[0, m-1]$.

Choice for n and internal state. The number of prime numbers n used in the internal state depends on the maximal possible depth for an address graph. For an address graph of size $\mathcal{N}$, the maximal depth is $\mathcal{N}-1$: $n \geq \mathcal{N}-1$. In [Zho08], the value $n = 209$ is used for $\mathcal{N} = 50$. It should be noticed that the initialization of the internal state of the root node (address a) has no impact on the respective period of the generator (see Equation 5).

To conclude, the Figure 9 shows several addresses assignment performed by Prophet for $\mathcal{N} = 12, n = 11, m = 13$ and different topologies.

6 Conclusion

We have seen the limits of the proposition of Prophet. Prophet is based on automaton which must be characterized by their inner behaviour (Assumption 1) and by their outer behaviour (Assumption 2). While the core idea of Prophet are sound, the application of pseudo-random generators leads to a scheme with many collisions. The scheme may works for some address graphs but many collisions can be expected for some setups. The key question of Prophet is how to design these functions in order to allow collision free address allocation.

References

[BCM09] C. Bernardos, M. Calderon, and H. Moustafa. Survey of IP address autoconfiguration mechanisms for MANETs. Technical report, IETF, 2009. `http://tools.ietf.org/html/draft-bernardos-manet-autoconf-survey-04`.

[Dur89] Mark J. Durst. Using linear congruential generators for parallel random number generation. In *21st conference on Winter simulation – WSC '89*, pages 462–466. ACM, 1989.

[Hal89] John H. Halton. Pseudo-random trees: multiple independent sequence generators for parallel and branching computations. *Journal Computational of Physics*, 84(1):1–56, 1989.

[Knu85] Donald E. Knuth. Deciphering a linear congruential encryption. *IEEE Transactions on Information Theory*, 31(1):49–52, 1985.

[Knu97] Donald E. Knuth. *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. Addison-Wesley, Inc., 1997.

[Leh51] Derrick H. Lehmer. Mathematical methods in large-scale computing units. In *Second Symposium on Large-Scale Digital Calculating Machinery*, pages 141–146. Harvard University Press, 1951.

[MDMD01] Archan Misra, Subir Das, Anthony McAuley, and Sajal K. Das. Autoconfiguration, registration, and mobility management for pervasive computing. *Personal Communications, IEEE*, 8(4), 2001.

[Nie92] Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*. Society for Industrial and Applied Mathematics, 1992.

[NP02] Sanket Nesargi and Ravi Prakash. Manetconf: configuration of hosts in a mobile ad hoc network. In *INFOCOM 2002*, volume 2, pages 1059–1068, 2002.

[PM88] Stephen K. Park and Keith W. Miller. Random number generators: good ones are hard to find. *Communication of the ACM*, 31(10):1192–1201, 1988.

[PRD00] Charles E. Perkins, Elizabeth M. Royer, and Samir R. Das. IP Address Autoconfiguration for Ad Hoc Networks. Technical report, IETF, 2000.

[Ste87] Jacques Stern. Secret linear congruential generators are not cryptographically secure. In *Symposium on Foundations of Computer Science - FOCS 1987*, pages 421–426. IEEE, 1987.

[Zho08] Hongbo Zhou. Secure Prophet Address Allocation for Mobile Ad Hoc Networks. In *IFIP International Conference on Network and Parallel Computing - NPC 2008*, pages 60–67, 2008.

[ZNM03] Hongbo Zhou, Lionel M. Ni, and Matt W. Mutka. Prophet address allocation for large scale manets. In *IEEE INFOCOM 2003*, 2003.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399