Cost-Optimization of the IPv4 Zeroconf Protocol

Henrik Bohnenkamp[*], Peter van der Stok[†], Holger Hermanns[‡,*], Frits Vaandrager[§]

[*]Dept. of Computer Science, University of Twente, Enschede, The Netherlands. Email: bohnenka@cs.utwente.nl

[†]Philips Research Laboratories, Eindhoven, The Netherlands. Email: Peter.van.der.Stok@philips.com

[‡]Dept. of Computer Science, Saarland University, Saarbrücken, Germany. Email: hermanns@cs.uni-sb.de

[§]Nijmeegs Inst. v. Informatica en Informatiekunde, University of Nijmegen, The Netherlands. Email: Frits.Vaandrager@cs.kun.nl

Abstract

This paper investigates the tradeoff between reliability and effectiveness for the IPv4 Zeroconf protocol, proposed by Cheshire/Adoba/Guttman in 2002, dedicated to the self-configuration of IP network interfaces. We develop a simple stochastic cost model of the protocol, where reliability is measured in terms of the probability to avoid an address collision after configuration, while effectiveness is viewed as the average penalty perceived by a user. We derive an analytical expression for the user penalty which we use to derive optimal configuration parameters of the network, restricting to those parameters which are under the control of a consumer electronics manufacturer. In particular we show that minimal cost and maximal reliability are qualities that cannot be achieved at the same time.

1 Introduction

Future generations of consumer electronic products such as DVD players, microwaves, TV-sets etc., are envisioned to be connected via a home local network, based on the Internet Protocol (IP) suite. Hand-held devices, laptops and wearable computers are additionally supposed to be able to form and communicate over so-called ad-hoc networks. To enable communication among appliances, each will be equipped with an IP network interface, making it possible to connect them as nodes to a network.

A prime requirement for the economical success of these types of networking is that their initialization and maintenance must rely on at most a minimal degree of user intervention. They must be *self-configuring*. When integrating a new appliance into an existing network, the requirement to be self configurable is of particular importance. Prior to being usable inside the network, the interface must be configured with a unique IP address. One solution to achieve this assumes a running (usually manually configured) DHCP server, which is responsible for attributing IP addresses dynamically. This is suboptimal for home networks, due to the need for manual intervention, and because this solution is not robust in case of a failure or unplugging of the DHCP server. In ad-hoc networks, a DHCP server is generally not even available. The ideal solution is a distributed "plug-and-play" solution, where the selection of a unique IP number is taken care of solely by the embedded control software of the individual appliance to be connected to the net.

In this paper, we study a simple protocol which has been proposed in the Internet-Draft [2] to perform the above "plug-and-play" task. This algorithm uses randomization to automatically configure an interface with an IPv4 address in the currently unassigned spectrum of addresses of the network. In the following, we address this protocol as the *zeroconf* protocol.

The major criterion for the assignment of IP numbers to interfaces is that the chosen IP number must be *unique* within a given network context. For ad-hoc networks and most home networks the network context can be considered to be link-local, *i.e.,* no routers are present within the network.[1] The Internet Assigned Number Authority (IANA) has allocated 65024 IP addresses for the purpose of communication between nodes (called hosts in the sequel) on a single link: spanned by the addresses 169.254.1.0 to 169.254.254.255. These IP-numbers are supposed to be used in a local network only, therefore they are never allowed to be routed.

The basic idea of the zeroconf protocol is easy to explain. A host that wants to configure a new IP link-local address randomly selects an IP address U out of the 65024 available names. It then broadcasts a message to the network "Who is using the address U?" We call such a message a *probe*. If a probe is received by a host that is already using address U, it will broadcast a reply indicating that U is in use. Upon receipt of this reply, the new host will start from scratch: it randomly selects a new address, broadcasts a new probe, etc. It may occur that a probe does not arrive due to message

[1]The link-local network can be connected to other IP nets via one or more routers, but link-local IP-addresses are not passed over these routers.

loss or a busy host, or that a reply gets lost. Therefore, to increase reliability, a host is required to send n probes, each time followed by a listening period of a certain length r. Only when during the total period of $n \cdot r$ seconds no reply message has been received, a host may start to use its new IP address. It is important to realize that when a host decides to use a new link-local IP address after sending four requests, it may still be possible that some other host in the network is using the same address, for instance, because all probes got lost. Such a situation, which is called *address collision*, may, in the worst case, force a host to kill active TCP/IP connections. This is highly undesirable.

The draft [2] suggests to set the length of the listening period to $r = 2$ seconds for unreliable (wireless) networks, and $r = 0.2$ seconds for reliable ones, but no precise argument justifying these values is given. The suggestion is based on assumptions of round-trip delays of the underlying physical network. Likewise, the number of probe transmissions is set to $n = 4$, but the influence of a variation of this number is not treated.

From a user perspective, it is desirable that the self-configuration of a device takes a minimal amount of time. For a hand-held device user, for instance, a configuration time of 8 seconds may seem barely acceptable, in particular if in combination with a perceived high risk of breaking existing connections, caused by address collisions.

On the other hand, decreasing the duration r of the listening periods may increase the probability of an address collision, and so does a decrease in the number n of probe transmissions: while sending less probes takes less time, it decreases the chance that a host will discover that an IP address is already in use. Thus, a trade-off has to be made by the manufacturer between the goal of reliably assigning a (locally) unique IP address, and the goal of not disturbing the user too much. This is the principal issue addressed in this paper, and we do so by addressing the following questions: "Is it actually needed to send $n = 4$ probes?"; "Are there variations of the protocol which behave equivalently except that configuration takes less time?"; "What is the probability that an address collision occurs in the initialization phase for a given variation of n and r?"; and "What is the optimal number of probes for a given scenario?"

The answers to these questions depend on the cost of having to wait versus the cost of address collisions. As a particularity of our approach, we treat costs as abstract, dimension-less entities which provide a common quantitative scale for very different aspects of user (dis)satisfaction such as experiencing waiting time as well as experiencing the consequences of an address collision (*i.e.,* broken connections). We model the initialization phase of the protocol as a stochastic cost model, namely a family of discrete-time Markov reward models. We provide an analytical evaluation of the model and address the above question based on our analytical insight. Our model abstracts away from many details, but allows us to exhibit the trade-off in the protocol design very clearly. The core of this approach is to find (i) the number of probes n that have maximally to be sent and (ii) the optimal length r of the waiting period such that the overall (mean) cost of the protocol is minimal. The model sets all important parameters and costs in relation to each other.

The benefits of our approach are the following: first, we can gain insight into the interplay between the different configuration parameters. Moreover, we can derive configuration parameters for the protocol, depending on the reliability of the underlying network technology and the cost of an address collision. Finally, we are able to assess the sensitivity of the measures of interest of our model to variations in the input parameters.

Related Work. In [7], a more detailed model of the zeroconf protocol is described and analyzed using the model checker Uppaal [1, 4]. In this work, the emphasis is on what happens in a setting in which multiple hosts simultaneously request an IP address. The analysis takes place in a setting of timed automata and does not take probabilistic aspects into account.

The rest of the paper is organized as follows. In Section 2, we describe the zeroconf protocol in greater detail. Section 3 introduces the model of the initialization mechanism of the protocol, while Section 4 derives an analytic cost function and evaluates it for specific scenarios. In Section 5 we discuss how the address collision probability can be obtained from the model. In Section 6 we give a final assessment of the protocol parameters chosen for the zeroconf protocol. Section 7 concludes the paper.

2 The IPv4 zeroconf protocol

This section describes the details of the zeroconf protocol as proposed in [2]. The core of the protocol comprises two parts. The first deals with the collision-avoiding assignment of an IP address to an interface during initialization, while the second part deals with the collision detection and address defense mechanisms during normal operation, which is a network maintenance task. Since the focus of this paper is on the initialization phase of the zeroconf protocol, we will describe only the first part.

We consider a fresh host h intending to connect to an existing IP network. First, the host h selects an IP address U randomly out of the reserved address space 169.254.1.0 to 169.254.254.255. Before h can use the address to configure its interface, it must achieve certainty that the address is not already in use by another host on the same link-local network. To find this out, host h uses the *address resolution protocol* (ARP), which is part of the IP suite [5]. The

ARP is used to find the hardware address of an interface connected to the link-local network that has been configured with a given IP address. When an arbitrary host on the network wants to determine the hardware address of a given IP address X, it sends out an ARP packet (which is broadcasted to every interface connected to the local link) containing X. This ARP packet is, figuratively speaking, the question "What is the hardware address that belongs to IP number X?", addressed to all other hosts on the link. The host with the interface configured with address X then sends a message to the querying host which contains the hardware address of the interface. If no host is configured with address X, there will be no answer.

The ARP mechanism is utilized by the zeroconf protocol to determine whether the chosen address U is already in use. Host h sends out so-called *ARP probes*. ARP probes are specially crafted ARP packets containing the randomly chosen IP number U. The question broadcasted on the net is then "What is the hardware address that belongs to IP number U?". Then, if some host h' has an interface configured with address U, it broadcasts the hardware address of its interface. In this case, however, the hardware address is not of interest, but only the fact that an answer has been sent. This is a clear indication for h that U is already in use and should not be used by the newcomer h. Then h must choose another address randomly and start anew. If, on the other hand, no answer has been received for U, this indicates that U is not yet in use. The protocol requires that n ARP probes have to be sent out by h, unless a response to at least one of them has been received. After each probe, r seconds have to elapse before the next probe is allowed to be sent. Consequently, before host h can configure its interface with an unused IP address, at least $n \cdot r$ seconds have to pass. [2] sets $n = 4$ and fixes $r \in \{2, 0.2\}$.

3 Modeling the zeroconf protocol

3.1 A family of discrete-time Markov reward models

The model of the zeroconf protocol is expressed in terms of a *family* of *discrete-time Markov reward models* (DRMs). A Markov reward model is basically a Markov chain equipped with a reward structure, the latter assigning rewards, bonuses, or dually costs, to the states and/or transitions of the chain. We consider the cost interpretation here. We deal with a *family* of DRMs, since we consider different Markov chains for different values of $n \in \{1, 2, 3, \ldots\}$, where n is the number of ARP probes needed to be sent before a decision of acceptance of the chosen IP address is made.

In our modeling efforts we focus on a model of a single representative host, which is assumed to be freshly connected to the local link. The link itself and all other hosts on the link are described in an abstract way: by probabilities. We assume that there are m hosts already connected to the network, and that, during the process of self-configuration of the host, other devices are neither added nor removed from the network, nor trying to acquire a new IP address.

The DRM family we consider is parametric in n, and its common structure is depicted in Figure 1. Costs are associated with transitions (given in parentheses in Figure 1; whenever no costs are indicated they are set to zero). We will now describe the model and the purpose of the costs in detail. Common to all of DRM are the states `start`, `er-`

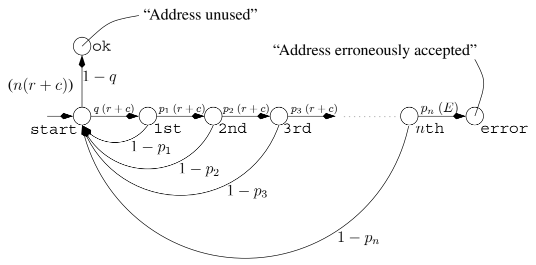

Figure 1. Structure of the DRM family

`ror`, and `ok`. Moreover, there are states `1st`, `2nd`, `3rd`, ..., n`th`. In the initial state, `start`, the host randomly selects an IP address from the provided address range. There are now two possibilities, which we model by two transitions outgoing from `start`. Either the IP address is already in use (represented by a transition to state `1st`), or the IP address is not in use yet (represented by a transition to state `ok`). The transition to `1st` has probability q, the one to `ok` $1-q$. The probability q depends on the number m of IP addresses in use by other hosts on the net. Assuming that each host possesses a single IP address we have $q = m/65024$.

State `ok` models the case that the chosen address is not in use yet. The host will then send n ARP probes, as specified by the protocol. Due to our assumption about the static nature of the network during the protocol run no host will send an ARP reply in response to these probes, so after $n \cdot r$ seconds the host may start using the new IP address. Sending an ARP probe and receiving a reply incurs some cost, for two reasons. (i) The network is a limited resource and its use should be accounted for. Therefore, for each ARP probe sent we charge a certain cost c. (ii) Since sending an ARP probe also means to wait for a certain time r, we also charge a cost of r to each ARP probe sent. Therefore, we attach the cost $n \cdot (r + c)$ to the transition from `start` to `ok`. If, on the other hand, the system moves to state `1st`, then the host will send an ARP probe, and listen for r seconds. To model the costs incurred, we attach the cost $r + c$ to the transition from state `start` to state `1st`. In this state, the host is asking to acquire an IP number which is already used by some other member of the network. But for different reasons, that member may not reply within r time units: either the probe got lost, or the host that ought to reply is busy, or the reply packet gets lost. To account for this, we assume that with a certain probability p_1 there will be no reply in response to the first ARP probe, and the protocol moves to state 2nd, in which a second ARP probe will be sent, which again costs $r + c$. If, however, all goes well the protocol will receive an ARP reply indicating that the selected IP address is already in use. In that case the model returns to the initial state start and starts all over again. This happens with probability $1 - p_1$. The behavior in states 2nd, 3rd, ..., nth is very similar to that in state 1st, only the probabilities of not receiving a reply to one of the previously sent out probes are $p_2, p_3, \ldots, p_n$. If in state nth the nth and last ARP probe still remains unanswered, the host will decide to start using the new IP address even though there is an address collision. In this case, the protocol has clearly reached an undesirable state, because the maintenance mechanism will later have to launch a costly protocol to re-establish the integrity of the IP numbers. Therefore we attach some (high) cost E to the transition from nth to error. This cost is an abstract value for the average burden incurred by the user due to the interrupt of the network service.

The above model abstracts away from a few details considered in [2]: (a) a host may decide not to retry IP addresses that failed before; (b) if the number of collisions a host has experienced so far exceeds 10 then it must limit the rate at which it probes for new addresses to maximally one address per minute.

3.2 No-answer probabilities

In the above description of the model the probabilities $p_1, p_2, \ldots, p_n$ were left unspecified. These probabilities, representing the probability of not receiving an answer in round $1, \ldots, n$ of the protocol have a decisive influence on the characteristics of the model, and we must provide reasonable values for them. In this section we will consider this problem.

There is certainly some interdependency between the probabilities p_i and r, the length of a listening period. For example, p_1 is the probability, that no reply to the first ARP probe sent arrives within the first listening period. Intuitively one would assume that, the shorter r is, the higher p_1, and the longer r, the lower p_1. If we assume a certain round-trip delay d for an underlying network, then we can be quite sure that $p_1 = 1$, if $r < d$. If $r > d$, then $p_1 < 1$ seems reasonable, but there might still be the possibility that the reply packet does not arrive.

A similar reasoning applies to p_2, *i.e.,* there might be a positive probability that the second ARP probe remains unanswered in the second listening period. However, it might be the case that the reply to the first ARP probe arrives in the second listening period, and this possibility must also be taken into account. So p_2 should be the probability that neither the reply to the second nor the reply to the first ARP probe is received in the second listening period. Consequently, p_3 must be the probability that no reply for the first, the second, or the third ARP probe is received in the third listening period, and so on for $p_4, p_5, \ldots, p_n$.

As a first conclusion, it is reasonable to assume that p_i is actually a function in r, $p_i(r)$. How do we define the $p_i(r)$? Consider first a random variable X which denotes the time that a reply to an ARP probe is received, once this probe has been sent. As usual, $F_X(t) = \Pr\{X \leq t\}$. We also know that $\Pr\{t_1 \leq X \leq t_2\} = F_X(t_2) - F_X(t_1)$. We define now the function $P : (\mathbb{N} \times \mathbb{R}^+) \to [0, 1]$ as:

$$P(i, r) = \prod_{j=1}^{i} \left(1 - \frac{F_X(jr) - F_X((j-1)r)}{1 - F_X((j-1)r)}\right), \qquad (1)$$

and we set $p_i(r) = P(i, r)$. A few remarks about (1) are in order. The expression $F_X(jr) - F_X((j-1)r)$ denotes the unconditional probability that a reply packet is received in the interval $[(j-1)r, jr)$. The quotient in the above equation is equal to the probability that a reply packet is received in the interval $[(j-1)r, jr)$, given that it has not yet arrived in the interval $[0, (j-1)r)$. We have to take this side-condition into account, since in state ith we know that no reply has arrived before. The complement $(1 - \cdots)$ appearing in the product in (1) is then the probability that the reply on the $(i - j + 1)$-th ARP probe does *not* arrive in interval $[(j-1)r, jr)$, given that it has not arrived in $[0, (j-1)r)$. The product over $j = 1, \ldots, i$ is then the probability that no reply to any of the i ARP probes sent out earlier arrives in interval $[(j-1)r, jr)$. Although it does not appear in the model, we define $p_0(r) = P(0, r) = 1$. Note that P is *not* a distribution function or density in r. Note further that by defining $P(i, r)$ as in (1), we silently assume that the ARP probes and the respective replies behave stochastically independent from each other. This is a simplification—usually this assumption is not justified, since error situations might have some persistence. As a consequence, the probability that a packet gets lost might increase in the case that the previous packet was lost (error bursts). Our model does not take this possibility into account.

Now, the question arises how F_X should be chosen. Preferably, it should be based on measurements. For now we do not have measurements available. However, we will later define distributions to demonstrate the concept. The distributions are not accurate in the sense that they describe the reality in detail, but they address an important issue, which we will explain now. Normally, distribution functions are monotonically increasing functions from 0 to 1, *i.e.,* if F is a distribution function, then $0 \leq F(0) \leq 1$, and $\lim_{t\to\infty} F(t) = 1$. However, if we assume that

$\lim_{t\to\infty} F_X = 1$, we implicitly assume that a reply to an ARP probe is only delayed (perhaps for a long time), but will always arrive eventually. The real loss of packets (perhaps caused by electro-magnetic interference on a radio based network) is not taken into account. A way to incorporate the possibility of packets losses is to consider *defective distributions*, *i.e.,* non-negative, monotonously increasing functions $D(t)$ such that $\lim_{t\to\infty} D(t) = l < 1$. Then $1-l$ is the probability that a reply is never received. Apparently, $F(t) = \frac{1}{l}D(t)$ is a distribution function, and $D(t) = lF(t)$ is the probability that a reply arrives *and* that it arrives in the interval $[0, t)$. We will see an example of such a distribution in Section 4.3.

3.3 Abstract costs

Our treatment of costs deserves some explicit discussion. From a certain point of view, we are facing three types of cost: *time*, *network usage*, and the *cost for an error* that is charged when an IP address is erroneously accepted. However, we deliberately blur such a threefold distinction, in order to provide a uniform way of modeling the customers perspective on each of the above. For us, cost is a quantity that weighs the influence of unwanted behavior on the user. Technically, a cost ζ is incurred if a transition with cost ζ is traversed. Each path through the DTMC has a so-called *total cost*, which is the sum of the costs of all transitions on the path.

The first type of cost in our model is *time*, more particularly, the time to wait for a response to an ARP probe. This is expressed by the cost parameter r in the model, and we assume a one-to-one correspondence between time and cost, *i.e.,* if the waiting time is two seconds, then we set $r = 2$.

As mentioned in Section 3.1, we must account for the network usage of the protocol, since it is a limited resource. We introduce therefore a second cost parameter c, which we call the postage for an ARP probe. It is difficult to quantify the postage in advance, but in Section 4.5 we will estimate a value for it.

The third type of cost is the cost E of erroneously accepting an IP address, even if it is already in use. This happens in case that all ARP probes remain unanswered. It is difficult to assign values for E a priori, since many different aspects play a role here. From a technical point of view, accepting an IP address that is in use requires eventually the reconfiguration and re-establishment of interfaces and connections, respectively, not only of the wrongly configured host, but perhaps even of the other host that got the IP number first. Other aspects are the user dissatisfaction with the product, when a reconfiguration becomes necessary. Despite the problem to quantify all these different "sources" of cost, we will estimate a value for E in Section 4.5.

4 The mean cost of a protocol run

Now that we have described our model of the zeroconf protocol, we can turn our attention to its analysis. Our central measure of interest is the mean total cost that is incurred during initialization, *i.e.,* on the way from state `start` to one of the absorbing states, `ok`, or `error`. Once we have a way to analyze this cost as a function $\mathcal{C}(n, r)$ in the parameters n and r, we are interested to determine values for the integer number n and the listening time r such that the mean total cost is minimal.

4.1 Cost function

Since the structure of the DRM family we consider possesses a simple repetitive structure, it is possible to derive an *analytic* expression that describes the mean total cost as a function $\mathcal{C}(n, r)$ in the parameters n and r and the coefficients $c,\ p_1(r), \ldots,\ p_n(r),\ q$, and E. This section discusses how we derive this cost function. We define the probability matrices $\mathbf{P}_n = (p_{ij}^{(n)})_{i,j=1,\ldots,n+3}$ and cost matrices $\mathbf{C}_n = (c_{ij}^{(n)})_{i,j=1,\ldots,n+3}$, for $n = 1, 2, \ldots$, where the matrix entries are defined as follows:

$$\begin{array}{lcll}
p_{1,2}^{(n)} & = & q & \\
p_{1,n+3}^{(n)} & = & 1-q & \\
p_{i1}^{(n)} & = & 1-p_{i-1}(r) & \text{for } i = 2,\ldots,n+1 \\
p_{i,i+1}^{(n)} & = & p_{i-1}(r) & \text{for } i = 2,\ldots,n+1 \\
p_{ii}^{(n)} & = & 1 & \text{for } i = n+2, n+3 \\
c_{1,n+3}^{(n)} & = & n\cdot(r+c) & \\
c_{i,i+1}^{(n)} & = & r+c & \text{for } i = 1,\ldots,n \\
c_{n+1,n+2}^{(n)} & = & E &
\end{array}$$

All other entries of $\mathbf{P}_n$ and $\mathbf{C}_n$ are zero. Note that the entries of both matrices depend on the length of the listening period r (*cf.* Section 3.2). The relation between states of the DRM and indices of the matrices is shown in the following table:

State	`start`	`1st`	$\cdots$	n`th`	`error`	`ok`
`row`$(\cdot)$	1	2	$\cdots$	n+1	n+2	n+3

Matrix $\mathbf{P}_n$ takes only probabilities into account, while matrix $\mathbf{C}_n$ describes the costs attached to transitions in the model. Note that, if $p_{ij} = 0$, then also $c_{ij} = 0$. Furthermore, we assure that $c_{ii} = 0$, for $i = n+2, n+3$. Otherwise the mean total cost would not be finite, because an absorbing state would allow to add costs unboundedly.

The mean total cost $\mathcal{C}(n, r)$ we are interested in is the one obtained when initialising the protocol in state `start`. In other words, it is the value of $a_1^{(n,r)}$, assuming a vector $\underline{a}' = (a_1^{(n,r)}, \ldots, a_{n+3}^{(n,r)})^T$ denoting the mean total costs for states $j \in \{1, \ldots, n+3\}$. The values of $a_i^{(n,r)}$ can be expressed as the solution to the following system of linear equations:

$$a_i^{(n,r)} = \sum_{j=1}^{n+3} p_{ij}^{(n)} \left(c_{ij}^{(n)} + a_j^{(n,r)} \right). \qquad (2)$$

The meaning of this system of equations is as follows: $c_{ij}^{(n)} + a_j^{(n,r)}$ is the cost of transition $i \to j$ plus the mean total cost of state j. So this is the total cost that incurs in case the transition $i \to j$ is chosen. This cost is weighted with the probability of actually taking this transition from state i, *i.e.,* $p_{ij}^{(n)}$. To obtain the mean total cost of state i we have to sum this quantity over all possible target states $j \in \{1, \ldots, n+3\}$. Apparently, $a_{n+2}^{(n,r)} = a_{n+3}^{(n,r)} = 0$.

We can rewrite the equation system of the form (2) in a single matrix-vector equation. Let $\mathbf{P}'_n = (p_{ij}^{(n)})_{i,j=1,\ldots,n+1}$ be the submatrix of $\mathbf{P}_n$ spanned by the non-absorbing states $1, \ldots, n+1$, and let $\underline{w} = (w_1, \ldots, w_{n+1})^T$ be a vector satisfying $w_i = \sum_{j=1}^{n+3} p_{ij} c_{ij}$. Then the vector $\underline{a}' = (a_1^{(n,r)}, \ldots, a_{n+1}^{(n,r)})^T$ of the mean accumulated costs for non-absorbing states $1, \ldots, n+1$ is the solution to the matrix equation $\underline{a}' = \mathbf{P}'_n \underline{a}' + \underline{w}$, or better, $\underline{a}' = -(\mathbf{P}'_n - \mathbf{I})^{-1} \underline{w}$. $\mathbf{P}'_n - \mathbf{I}$ is regular, which follows from the Perron-Frobenius-Theorem for decomposable stochastic matrices [6].

For our particular case, we are actually only interested in $\mathcal{C}(n,r) = a_1^{(n,r)}$, the mean accumulated cost of the starting state `start`. Due to the simple structures of the matrices involved, we can derive a symbolic solution for $\mathcal{C}(n,r)$:

$$\mathcal{C}(n,r) = \frac{(r+c)\left(n(1-q) + q \sum_{i=0}^{n-1} \pi_i(r)\right) + qE\pi_n(r)}{1 - q(1-\pi_n(r))}, \qquad (3)$$

where $\pi_i(r) = \prod_{j=0}^{i} p_j(r)$, for $i = 0, \ldots, n$. Recall from Section 3.2 that the $\pi_i(r)$ are determined by the defective probability distribution F_X describing the time to receive a reply to an ARP probe.

4.2 Finding parameters for minimal cost

Equation (3) provides us with the principal means to look into all the facets of $\mathcal{C}(n,r)$, the mean total initialization cost of the IPv4 zeroconf protocol. It shows that $\mathcal{C}$ depends on two types of parameters, namely the explicit parameters n and r which are under the control of the designers of the protocol, and the application specific parameters c, q, E, and F_X which can be predicted by the protocol designers to only a very limited extent.

We can now identify two different ways to study the function $\mathcal{C}(n,r)$. One may either fix the protocol by fixing n and r, and then perform a sensitivity analysis with respect to one of the remaining, application specific, parameters. Or one may fix an application scenario by fixing c, q, E, and F_X, and strive for an optimal setting of the protocol parameters n and r. The earlier approach is a standard exercise, and we return to this strategy in Section 4.5 where we discuss the parameters c and E, and in Section 6. Here we instead focus on the optimization aspect. We formulate the optimization problem as follows. For fixed c, q, E, and F_X, we intend to *find the pairs* $(n,r) \in \mathbb{N} \times \mathbb{R}^+$ *such that* $\mathcal{C}(n,r)$ *is minimal.*

The parameter n is discrete, and thus we can define $\mathcal{C}(n,r)$ as a family of functions $\{\mathcal{C}_n\}$, where $\mathcal{C}_n(r) := \mathcal{C}(n,r)$. We are interested in the shape of $\mathcal{C}_n$, describing the mean cost as a function of r when n has been fixed. The first question of interest concerns the shape $\mathcal{C}_n$ of this function. Due to our assumption of a defective distribution F_X we have $\lim_{r\to\infty} F_X(r) = l$. It is not difficult to verify that in this case $\pi_i(0) = 1$ and that $\lim_{r\to\infty} \pi_i(r) = (1-l)^i$, for $i = 1, \ldots, n$. Moreover, $\pi_n(r)$ falls off polynomially with degree n. As a consequence, $\mathcal{C}_n(0) = qE$, and for $r \to \infty$, $\mathcal{C}_n(r)$ is approaching the asymptote

$$\mathcal{A}_n(r) = \frac{(r+c)\left(n(1-q) + q\left(\frac{1-(1-l)^n}{l}\right)\right)}{1-q}.$$

Apparently, the cost function $\mathcal{C}_n$ is a mixture of the linearly increasing function $\mathcal{A}_n$ and the polynomially decreasing $qE\pi_n(r)$. As long as we can assume that $\mathcal{A}_n(r) << qE$ for small r (which is realistic, since we assume E to be rather large), we can also assume that $\mathcal{C}_n(r)$ first falls off polynomially to a minimum, and increases then linearly, as r increases.

To address the above optimization problem we are searching for the number $r_{\text{opt}}^{(n)}$ such that $\mathcal{C}_n(r_{\text{opt}}^{(n)})$ is minimal. Computing $r_{\text{opt}}^{(n)}$ is best done by numerical means. It is possible to use the derivation of the cost function and to derive the zeroes, or to compute the minimum by some other means. From a numerical point of view this is not particularly challenging, and we will therefore not dwell on the solution aspects of the model. All numeric results and example plots in this paper have been computed numerically by means of the Maple[2] tool.

4.3 Example plots

In this section, we will show some sample graphs for $\mathcal{C}_n$. To do so, we first have to fix the parameters c, q, E, as well as the probability distribution F_X (*cf.* Section 3.2) for the time between sending an ARP probe and receiving a reply. As mentioned earlier, F_X should be based on measured data. Since we do not heave measures at hand, and we want to demonstrate the principle rather than to produce exact results in a particular setting, we decide to define F_X on the basis of an exponential distribution.

$$F_X(r) = \begin{cases} l \cdot \left(1 - \mathrm{e}^{-\lambda(r-d)}\right) & \text{for } r \geq d \\ 0 & \text{otherwise,} \end{cases}$$

[2]Maple is a registered trademark of Waterloo Maple Inc.

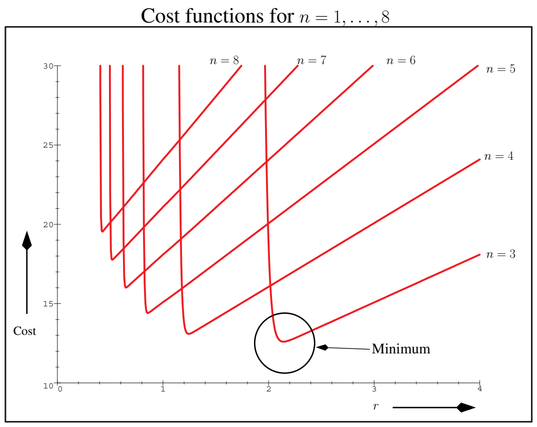

Figure 2. Cost functions $\mathcal{C}_1, \ldots, \mathcal{C}_8$

where $1 - l$ is the probability that an ARP probe never gets a reply, d is the round-trip delay of the considered network, and $d+1/\lambda$ is the mean time a reply is received after sending the ARP probe, assuming that the reply does not get lost. In the following, we choose $d = 1$, $l = 1-10^{-15}$, and $\lambda = 10$. For the probability q that an IP address is in use we choose $q = 1000/65024$. For the cost of sending an ARP probe we assume $c = 2$, and for E, the cost of an erroneously accepted IP address, we choose $E = 10^{35}$. In Figure 2, we show some examples plots of cost functions $\mathcal{C}_n(r)$ for $n = 1, \ldots, 8$. Actually, the functions for $n = 1, 2$ are not visible, since their smallest values are much too large to fit into the chosen range of the plot. As we can see, the cost functions indeed all have a minimum. The higher n is chosen, the smaller $r_{\text{opt}}^{(n)}$. However, $\mathcal{C}_3(r_{\text{opt}}^{(3)}) < \cdots < \mathcal{C}_8(r_{\text{opt}}^{(8)}) < \cdots$. The increase of the minimum of the cost function $\mathcal{C}_n(r)$ for larger n is mainly caused by the postage c. If we would set $c = 0$, then the optimal strategy would be to send as many ARP probes as fast as possible, without waiting for a reply. Then the probability that at least one reply returns d seconds (the roundtrip delay) after sending the first ARP probe approaches one. Since in the real world the sending of a packet incurs some nonzero cost, this strategy is not recommendable.

4.4 Optimal n

Up till now, we have only considered the optimal value of r for a particular n. In this section, we want to address the question what n should be chosen for a given r, such that the cost is minimal. Therefore, we now define a function $N : \mathbb{R} \longrightarrow \mathbb{N}$, that returns the optimal number n that minimizes $C(n, r)$ for fixed r: $N(r) = \min\{\, n \in \mathbb{N} \mid \mathcal{C}_n(r) = \inf_{k\in\mathbb{N}}\{\mathcal{C}_k(r)\}\}$.

By means of $N(r)$ we can now define $\mathcal{C}_{\min}(r) = \mathcal{C}(N(r), r)$. $\mathcal{C}_{\min}$ describes the total cost of a protocol run under the condition that always an optimal n has been chosen for a given r. Figure 4 depicts a graph of $\mathcal{C}_{\min}$, based on the same parameters chosen for Figure 2. $\mathcal{C}_{\min}$ is described by the lower edge of the union of all function graphs of the $\mathcal{C}_n$. From the cost function, a rough estimate for the number

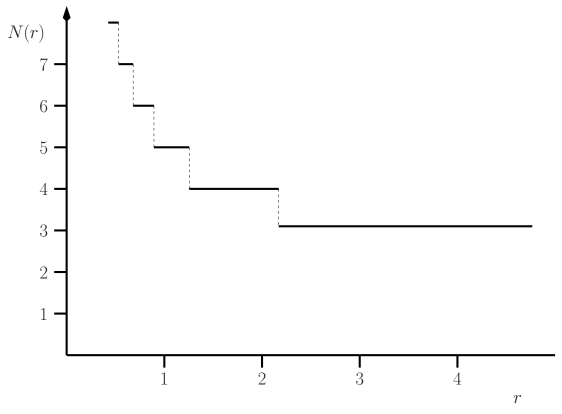

Figure 3. $N(r)$: optimal n for given r

$N(r)$ can be derived. As we have seen above, the cost function is a mixture of a linearly increasing and a polynomially decreasing function in r. To have the costs small, the polynomial part (*i.e.,* $qE\pi_n(r)$) should approach a number close to zero for increasing r. This can however only be achieved, if the influence of the cost parameter E is dwarfed by $\pi_n(r)$. As $\lim_{r\to\infty} \pi_n(r) = (1-l)^n$, we have that $(1-l)^n qE$ should be near to zero. Therefore, the minimal value of n is now

$$\nu = \left\lceil -\frac{\log(E)}{\log(1-l)} \right\rceil.$$

For all $n' < \nu$ we can be sure that $qE\pi_{n'}(r)$ will never approach zero. We now have another explanation why the graphs for $n = 1, 2$ are not visible in Figure 2: since $E = 10^{35}$ and $1 - l = 10^{-15}$, we have $\left\lceil -\frac{\log(E)}{\log(1-l)} \right\rceil = 3$, and therefore, it is impossible to achieve a reasonable cost, if $n = 1, 2$.

4.5 Cost of error

In [2], several assumptions are made about the parameters of the protocol. The maximal number n of ARP probes is set to $n = 4$. The waiting time r between the probes is set to $r = 2$ or $r = 0.2$. The round-trip delay is assumed to be maximally one second, the network speed minimally 1 Mbit/sec. No assumptions are made about the expected number of hosts on the link. We must assume that the chosen parameters cover the worst case with respect to speed, reliability, network size, and traffic. Up till now we have only assumed arbitrary values for the cost variable E.

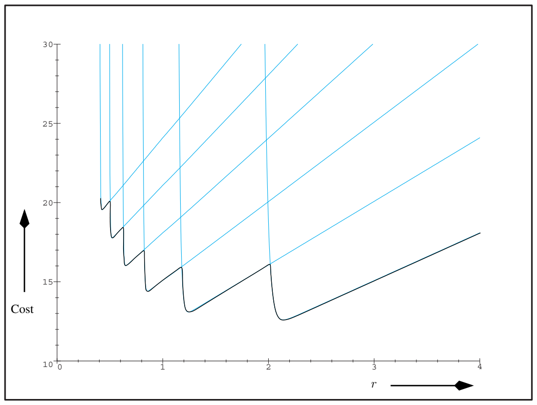

Figure 4. The minimal-cost function $\mathcal{C}_{\min}(r)$

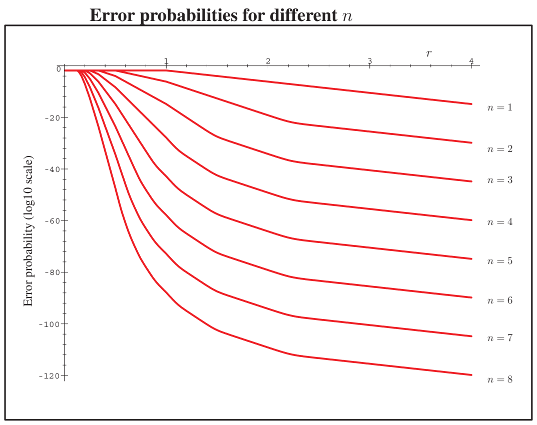

Figure 5. Probability to reach state `error`

An interesting question is what values the cost parameter E and the postage c must have such that the assumptions made in [2] become reasonable, *i.e.,* that $n = 4$ and $r = 2$ ($r = 0.2$) are the optimal choice with respect to cost minimization. This is addressed as follows. We choose very pessimistic parameters for l and q, and compute E and c for both values of r, which we will denote by $E_{r=2}$, $E_{r=0.2}$, $c_{r=2}$, and $c_{r=0.2}$. For the loss probability l we choose $l = 1 - 10^{-5}$, which for modern networks is very high. We assume that 1000 hosts are already connected to the network, *i.e.,* $q = 1000/65024$, and a worst-case round-trip delay of 1 second, *i.e.,* $d = 1$. The mean time until a reply is received after an ARP probe has been sent is $d + 1/\lambda = 1.1$, *i.e.,* $\lambda = 10$. For these parameters we can derive $E_{r=2} = 5 \cdot 10^{20}$ and $c_{r=2} = 3.5$ (by simple numerical approximation).

For $r = 0.2$ we must first realize that assuming a round-trip delay of 1 second is not reasonable, since with $n = 4$, the overall listening time has a duration of only 0.8 seconds. Therefore, shorter round-trip delays must be considered. We therefore assume now $d = 0.1$, which is still very pessimistic for a local net. We then also adapt the mean time until a reply is received to $d + 0.01 = d + 1/\lambda$, *i.e.,* $\lambda = 100$. Moreover, in [2] it is stated that choosing $r = 0.2$ is only permitted "*when the link [...] can be reasonably trusted to deliver packets reliably*". We interpret this as a hint to lower the loss probability. We assume therefore, that $1 - l = 10^{-10}$, which still is pessimistic. For the chosen values we can derive that $E_{r=0.2} = 10^{35}$ and $c_{r=0.2} = 0.5$.

5 Reliability

In this section, we will consider the reliability of the zeroconf protocol. We consider the reliability as the probability that an previously unused IP number has been selected once the initialization phase has terminated. In terms of our model this is given by the probability to end up in state `ok` while the probability to end up in state `error` corresponds to the complementary outcome, where the chosen number is already in use.

Computing these probabilities is a standard problem for discrete-time Markov chains, and the solution is described in detail in, for example, [3]. Therefore, in the following, we describe only briefly how to compute them.

Let the matrix $\mathbf{P}_n$ be as in Section 4.1, We partition this matrix as follows. Let $\mathbf{P}'_n = (p^{(n)}_{ij})_{i,j=1,\ldots,n+1}$, *i.e.,* $\mathbf{P}'_n$ is obtained from $\mathbf{P}_n$ by deleting the right-most two columns and the bottom-most two rows. Let $\underline{e}_n = (p^{(n)}_{1,n+2}, \ldots, p^{(n)}_{n+1,n+2})^T$, and $\underline{o}_n = (p^{(n)}_{1,n+3}, \ldots, p^{(n)}_{n+1,n+3})^T$ be the two vectors describing the one-step absorption probabilities into the state `error` ($\underline{e}_n$), respectively `ok` ($\underline{o}_n$). The matrix $\mathbf{P}_n$ can be reassembled from these fragments as follows:

$$\mathbf{P}_n = \left(\begin{array}{c|c|c} \mathbf{P}'_n & \underline{e}_n & \underline{o}_n \\ \hline \underline{0} & 1 & 0 \\ \hline \underline{0} & 0 & 1 \end{array}\right).$$

Since $\mathbf{P}_n$ describes the one-step transition probabilities in the Markov chain, it's k-th power describes the k-step probabilities, *i.e.,* the probabilities to jump from state i to state j in exactly k steps. We are now interested in the probabilities to end up in one of the absorbing states `error` or `ok`, given that we start in state `start`, in any number of steps. Let $\underline{s}_n = (1, 0, \ldots, 0) \in \mathrm{I\!R}^{n+1}$. We can express the probability to jump to, say, state `error` in exactly k steps, for $k = 1, 2, 3, \ldots$, when starting in state `start`, as $\underline{s}_n(\mathbf{P}'_n)^{k-1}\underline{e}_n$. We are looking for the probability $\mathcal{E}(n, r)$ to jump to state `error` in any number of steps, which is

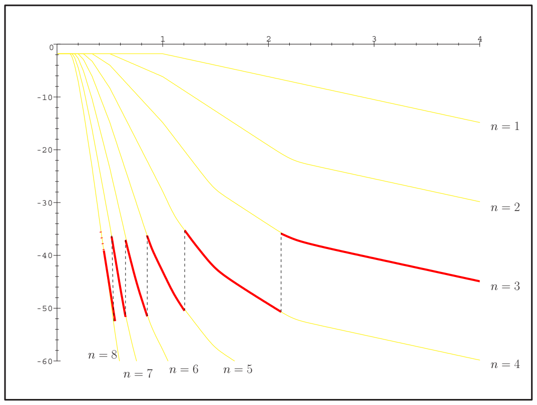

Figure 6. Error probability under optimal cost $\mathcal{E}(N(r), r)$

then $\underline{s}_n \sum_{k=1}^{\infty} (\mathbf{P}'_n)^{k-1} \underline{e}_n = \underline{s}_n (\mathbf{I} - \mathbf{P}'_n)^{-1} \underline{e}_n$. The complementary probability to reach state `ok` can be computed in the same fashion by replacing $\underline{e}_n$ by $\underline{o}_n$, or directly by taking the complement $1-\mathcal{E}(n, r)$.

As in Section 4.1, the simple matrix structure of $\mathbf{P}'_n$ allows us to derive an analytic expression for the probability $\mathcal{E}(n, r)$ given by

$$\mathcal{E}(n, r) = \frac{q\pi_n(r)}{1 - q(1 - \pi_n(r))}, \tag{4}$$

where $\pi_n(r)$ are the quantities appearing in equation (3). To give some insight into the above equation, we first interpret the denominator. If starting in state `1st`, the probability to go directly (*i.e.,* in n steps) to state `error` is equal to $\pi_n(r)$. Thus, the complement probability $1 - \pi_n(r)$ is the probability to take any path, except the one directly to state `error`. $q(1 - \pi_n(r))$ is then the probability to start in state `start`, go in one step to state `1st`, and to take any path then, except the direct one to state `error`. So the denominator $1 - q(1 - \pi_n(r))$ in equation (4) describes apparently the probability to start in state `start`, and to go directly either to state `ok` or to state `error`. We denote this event as B. The enumerator of (4) is the probability to go directly to state `error`, starting from state `start`. We denote this event as A. Then $\mathcal{E}(n, r)$ is the conditional probability $\Pr\{A|B\}$ to reach state `error` from state `start` under the condition that one of them is reached on a direct way. One might wonder if not also the detours should be taken into account, *i.e.,* all paths which touch the state `start` more than once. However, the probabilities to reach state `start` a second, third, fourth, ... time again sum up to one and, therefore, do not contribute to the outcome.

We have again derived an analytic expression, this time to express the probability to reach an error state, which is the complement of the protocol reliability. In Figure 5, we see examples for the probability $\mathcal{E}(n, r)$ for $n = 1, \ldots, 8$, where the probability to reach state `error` is plotted against the value of r. Note that the probability axis has a logarithmic scale.

One may study the behavior of $\mathcal{E}$ in multiple ways. Here, we combine it with our cost analysis of the protocol, and focus on the fragment of the parameter range of n and r that gives optimal total costs. To do so, we consider the function $\mathcal{E}(N(r), r)$, *i.e.,* we consider the case that the number of ARP probes is always chosen with optimal total costs, depending on r (cf. Section 4.4). In Figure 6, we see a sample plot for $\mathcal{E}$, embedded in the original graph of Figure 5. We see that the plot of $\mathcal{E}(N(r), r)$ has a peculiar shape. The most important feature is that $\mathcal{E}(N(r), r)$ has several maxima, which correspond to the steps of the piecewise constant function $N(r)$, and is piecewise continuously decreasing between these steps. To explain this shape, let (a, b) be an interval of maximal size where $N(r) = k$ for all $r \in [a, b]$. It is not surprising that $\mathcal{E}(N(r), r)$ is continuously decreasing in (a, b), since the waiting time r is increasing and influences the probability that a reply is received in the listening period to the better. The number of ARP probes, however, remains constant, since it is a discrete quantity. Only at the jump at b, $N(b+) = k - 1$, *i.e.,* the optimal number of ARP probes to be sent is decremented by 1. This has a negative effect on the error probability again, since now there is one chance less to send an ARP probe, and in particular only a shorter overall time to wait for replies on earlier ARP probes sent. This results results in a sharp increase at b. In fact, since $k - 1$ is the cost-optimal number of ARP probes to be sent and b is the smallest value for which this optimum holds, this induces a local maximum for the error function at b.

We observe that the minima of the cost function (Figure 4) do not correspond to the minima of the error function $\mathcal{E}(N(r), r))$. Moreover, the lower the probability, the higher the cost, as can be seen by comparing Figure 4 and 6. This indicates that optimal reliability and optimal cost can not be achieved at the same time. However, we can observe that the error is bounded and stays roughly within the limits of $[10^{-35}, 10^{-54}]$. So, one may draw the conclusion that, at least for this application scenario and similar ones, the trade-off between reliability and cost does exist, but is not particularly substantial. Even though optimal cost implies sub-optimal reliability, the latter is still very low in all cases we considered.

6 Assessing the IPv4 protocol

Now that we have derived the apparatus to describe the mean cost of a protocol run and the error probabilities, we can come back to assess the chosen parameters proposed in [2]. In Section 4.5, we have derived values for E and c, such that the proposed parameters $n = 4$ and $r = 2$ ($r = 0.2$) yield minimal cost, assuming a worst-case scenario. It is now interesting to observe how n and r change, if we assume a bit more optimistic scenarios. The only parameters we want to keep constant are E, c, and q. All other parameters are subject to change.

We first assume that we have a very reliable network with a loss probability of $1 - l = 10^{-12}$ (which is met by most modern ethernets). We also assume that the round-trip delay is small: a realistic value is $d = 1$ms. For these values we find out that the optimal parameters are $n = 2$, and $r \approx 1.75$. For these values, the probability that an address has been erroneously accepted is $\mathcal{E}(2, 1.75) \approx 4 \cdot 10^{-22}$.

We see that in case of more realistic parameters the duration of a listening period and especially the number of ARP probes sent can be chosen much smaller. So for the current case, the waiting time will be generally only about 3.5 seconds, rather than 8. Assuming less than $m = 1000$ hosts will also allow one to drop the waiting time and thus the total costs further.

7 Conclusion

This paper has presented a study of the IPv4 zeroconf protocol, an upcoming protocol targeting at the autonomous configuration of network interfaces with unique IP numbers at startup time.

We have presented a quantitative cost model of the IPv4 zeroconf protocol, defined as a family of simple DRMs. We decided to use an abstract notion of costs represented as dimension-less entities. This notion provides a common quantitative scale for very different aspects of user (dis)satisfaction. In the current consumer electronic market, efforts to decrease the mean dissatisfaction (or complementary, and in jargon, to increase the "fun-factor") of an electronic device is a prime design goal, but difficult to manage or optimize during the design cycle. Our study of the mean total cost of the protocol is precisely targeting in this direction.

The model enabled us to study analytically both the mean total cost of running the protocol, as well as its reliability. By varying the various parameters of the model we were able to put both reliability and cost in relation to each other. The main emphasis has been to isolate optimal values for those parameters that are under the control of the protocol designer, namely the number n of ARP probes to be sent, and the length r of the listening time needed to expire after sending each ARP probe. We have seen that minimal cost and minimal error propability is something that can not be achieved at the same time. In a nutshell, the lower r is set, the lower the cost become, but also the reliability decreases then.

The cost model we introduced is small and abstract. Nonetheless it provides already genuine insight in the mechanisms of the protocol. The numerical computations to derive the results from the model are very simple (computing the minima of functions), therefore, it should be possible to concretize the model, and keep the numerical derivations still feasible.

Another aspect to mention is that the application area of this protocol is potentially very broad, and the whole area of ad-hoc networks is very young and developing further with enormous speed. This is problematic, because the quality of the optimized protocol parameters depends to some extent on the quality of application specific parameters (such as the message loss probability) fed into the model. These parameters must be based on measurement in real world scenarios. This however is a delicate task for the designers, since they are developping their products for a future application profiles which are difficult to predict in the required degree of detail today. A certain flair is thus indispensable, but the analytical functions we provide are effective means to show the influence of such design decisions in any case.

Acknowledgments. We thank Miaomiao Zhang and Joost-Pieter Katoen for valuable comments on this paper. This work is supported by STW/PROGRESS project TES4999: Verification of Hard and Softly Timed Systems (HaaST), and the DFG/NWO bilateral cooperation project 600.050.011.01: Validation of Stochastic Systems (VOSS).

References

[1] G. Behrmann, A. David, K. G. Larsen, O. Möller, P. Pettersson, and W. Yi. UPPAAL - present and future. In *Proc. of 40th IEEE Conference on Decision and Control*. IEEE Computer Society Press, 2001.

[2] S. Cheshire, B. Adoba, and E. Guttman. Dynamic configuration of IPv4 link-local addresses. `http://www.ietf.org/internet-drafts/-draft-ietf-zeroconf-ipv4-linklocal-07.txt`, August 2002. DRAFT.

[3] V. G. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, London, Glasgow, Weinheim, 1995.

[4] P. Pettersson and K. G. Larsen. UPPAAL2k. *Bulletin of the European Association for Theoretical Computer Science*, 70:40–44, Feb. 2000.

[5] D. C. Plummer. An ethernet address resolution protocol, November 1982. Internet Standard 37, RFC 826.

[6] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[7] M. Zhang and F. Vaandrager. Analysis of a protocol for dynamic configuration of IPv4 link local addresses using Uppaal. Report NIII-R03XX, Nijmeegs Instituut voor Informatica en Informatiekunde, University of Nijmegen, 2003. To appear.