

Cost-Optimisation of the IPv4 Zeroconf Protocol

H. Bohnenkamp H. Hermanns

University of Twente

P.O. Box 217, 7500 AE Enschede, The Netherlands

Phone: +31 53 489 4662 Fax: +31 53 489 3247

Email: {bohenka|hermanns}@cs.utwente.nl

M. Zhang F. Vaandrager

University of Nijmegen

P.O. Box 9010, 6500 GL Nijmegen The Netherlands

Phone: +31 24 365 2216 Fax: +31 24 365 3137

Email: {miaomiao|frits.vaandrager}@cs.kun.nl

Abstract— We develop a stochastic model of a simple protocol for the self-configuration of IP network interfaces. We describe the mean cost that incurs during a self-configuration phase and describe a trade-off between reliability and speed. We derive a cost function which we use to derive optimal parameters. We show that optimal cost and optimal reliability are qualities that cannot be achieved at the same time.

Keywords— Embedded control software; IP; zeroconf protocol; cost optimisation

I. INTRODUCTION

Coming generations of consumer electronic products like VCRs, TV-sets, microwave ovens etc., are envisioned to be connected to a home local network, based on the Internet Protocol (IP) suite. The appliance is supposed to have a network interface over which communication with other appliances takes place. Before an interface is usable, it must be configured with an IP address. The user of such an appliance can however not be bothered with the configuration of the network interfaces, since it requires some technical knowledge that he cannot be assumed to have. Also DHCP servers are out of the question, since the setup of such servers would require at least a comparable technical understanding of the matters. Ideal would be a “plug-and-play” solution, where the configuration of the interface with an IP number would be taken care of by the embedded control software of the appliance.

A major criterion for the assignment of IP numbers to interfaces is that the IP number is *unique* on the network. Since home local networks are considered to be link-local, *i.e.*, not connected to other IP nets by means of routers, the uniqueness can be achieved for self-configuring interfaces. The Internet Assigned Number Authority (IANA) has allocated 65024 IP addresses for this purpose, *i.e.*, for communication between hosts on a single link: the prefix 169.254/16, except for the first and last 256 addresses. In this paper, we study a simple protocol that has been proposed in the Internet-Draft [1], by which a host may automatically configure an interface with an IPv4 address in the 169.254/16 address range that is unused on the local link.

The basic idea of the protocol is easy to explain. A host that wants to configure a new IP link-local address randomly selects an IP address $u.v.w.x$ out of the 65024 available names. It then broadcasts a message to the network “May I use the address $u.v.w.x$?” We call such a message a *probe*. If a probe is received by a host that is already using the address, then this host will send a reply “No!”. Upon receipt of a 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 loss or a busy host, or that a reply gets lost. Therefore, to ensure reliability, a host is required to send *four* requests, each time followed by a 2 second wait (in [1] a round-trip latency of at most one second is assumed). Only after the total period of 8 seconds has expired and no reply message has been received, a host may start to use its new IP address. It is important to realise that when a host decides to use a new link-local IP address after sending four requests, it is still 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 worst case, force a host to kill active TCP/IP connections. This is highly undesirable.

For consumer electronic appliances, a configuration time of at least 8 seconds is quite long. Users are expecting their devices to react instantly on their command, otherwise they lose faith in its serviceability. Therefore, we address in this paper the following questions: “Is it actually needed to send *four* probes?”, “Are there variations of the protocol which are equivalent except that configuration goes faster?”, and “What is the probability that an address collision occurs in the initialisation phase?”

In the design of the protocol, a trade-off had to be made between the primary goal of assigning a (locally) unique IP address, and the secondary goal of not using too much time for achieving this. Sending more requests takes time, but increases the chance that a host will discover that an IP address is already in use. The answer to the question what the optimal number of probes is depends on the cost of having to wait versus the cost of address collisions. Below,

we present a very simple model of the protocol for which we can fully analyse this optimisation problem. Our model abstracts away from many important 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 that have maximally to be sent and (ii) the optimal length of the waiting period such that the overall (mean) cost of the protocol is minimal.

We will develop a simple probabilistic stochastic model of the initialisation phase of the protocol which sets all important parameters and costs in relation to each other. The benefits of our approach are the following: first, we can gain insight in the interdependency 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 outcomes to variations in the input parameters.

The rest of the paper is organised as follows. In Section II, we describe the zeroconf protocol in greater detail. In Section III, we introduce the model that we have made of the initialisation mechanism of the protocol. In Section IV, we describe the methods used to obtain numerical results from the model, and discuss the results obtained from two examples. In Section V, we study the interdependency of the number of probes sent, and the time to wait for a reply. Finally, in Section VI, we conclude the paper.

II. THE IPV4 ZEROCONF PROTOCOL

In this section, we will describe the zeroconf protocol, as proposed in [1], in greater detail. The core of the protocol comprises two parts. The first deals with the collision-free assignment of an IP address to an interface during the interface initialisation, whereas the second deals with the collision detection and address defence during normal operation. Since we want to consider only the first part of the protocol in this paper, we will treat only this in detail in Section II-A. The second part of the protocol is described very concisely in Section II-B.

A. Selecting and Claiming an Address

A host h that wishes to configure an interface automatically chooses 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 make sure that the address is not already in use by another host on the same link-local network. The link-local network of h comprises all those hosts (including h) that can communicate with each other by means of the IP protocol without

the help of an IP router.

To determine whether the chosen address is in use, host h makes use of the *address resolution protocol* (ARP), which is part of the IP suite. 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 address¹. 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 net. The host with the interface configured with address X then returns the hardware address of the interface to the enquiring host². If no such host exists, there will be no answer.

The ARP mechanism is utilised 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 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 sends an answer back to h , providing the hardware address of the interface. In this case, however, the hardware address is not of interest, but only the fact that an answer has been returned. This is a clear indication that U is already in use and should not be used by h . In that case 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 four ARP probes have to be sent out by h , unless a response to one of them has been received already. After each probe, two seconds have to elapse before the next probe is to be sent. Consequently, before host h can configure its interface with an unused IP address, at least eight seconds have to pass.

B. Defending an Address

Once an interface is up and running, it continuously has to check whether an address collision has occurred, by whatever cause. In [1], two ways to react are provided when an address collision has been detected: either the host gives up its own address immediately and chooses a new one; or it defends the own address (in case that there are already some open TCP connections, for example).

¹The hardware address is needed to ensure that IP packets reach their destination.

²Usually, the IP-to-hardware address mapping is cached by a host to reduce the number of ARP requests. This detail plays here however no role.

The action taken to defend the own address X is simply to send an ARP message on the network that means, figuratively, “Address X is mine!”. Then normal operation continues. If the address collision has not been resolved, this will be eventually detected by the collision monitor. In that case, the host has to give up its address immediately.

III. MODELLING THE ZEROCONF PROTOCOL

Our model of the initialisation phase is expressed in terms of a *discrete-time Markov chain*.

The model that we propose is depicted in Figure 1. It

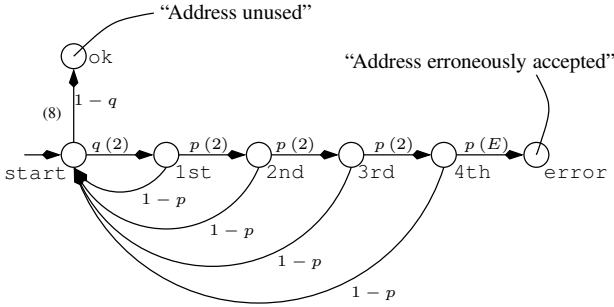


Fig. 1
SIMPLE MODEL

is a simple discrete-time Markov chain (DTMC) in which costs are associated to transitions (given in parentheses in Figure 1; whenever no costs are indicated it is assumed to be zero). In the initial state, *start*, a host randomly selects a new IP address. There are now two possibilities, which we model by two transitions outgoing from state *start*: either the IP address is already in use by some other host connected to the network. This is denoted by the transition to state *1st*. Otherwise, the IP address is not in use; this is denoted by the 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 local link, *i.e.*, $q = m/65024$. We assume that, as long as the self-configuration takes, no other devices will be added or removed from the network, or try to acquire a new IP addresses. The value m is a model parameter.

In state *ok* the host will send 4 ARP probes, as specified by the protocol. Due to our assumption about the static nature of the network, we know for sure that no one will send an ARP reply in response to these probes, so that after 8 seconds the host may start using the new IP address. We therefore attach the cost 8 to the transition from *start* to *ok*. If the system moves to state *1st*, then the host will send a first ARP probe followed by a two second wait. To model this delay, we attach the cost 2 to the transition from

state *start* to state *1st*. We assume that with probability p there will be no reply in response to the first ARP probe: either the probe gets lost, or the host that ought to reply is busy, or the reply packet gets lost. Therefore, with probability p the protocol moves to state *2nd*, in which a second ARP probe will be sent. In state *1st* with probability $1-p$ 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. Here our model abstracts away from the fact that in reality:

- a host will remember the IP address that failed and never try it again;
- a host should maintain a counter of the number of address collisions it has experienced in the process of trying to acquire an address, and if the number of collisions exceeds 10 then the host must limit the rate at which it probes for new addresses to no more than one address per minute. The behaviour in states *2nd*, *3rd*, and *4th* is very similar to that in state *1st*. In state *4th* the host has sent 4 ARP probes. If the fourth and last ARP probe gets lost, 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 and therefore we attach a cost (some big number) E to the transition from *4th* to *error*. Thus, the total cost of an execution in our model is the sum of the total waiting time and E (provided an address collision occurs).

The values of p and E are also model parameters.

IV. ANALYSING THE MODEL

In this section, we describe how we can compute the mean costs that accumulate during the period from system *start* until absorption in either state *ok* or *error*. We will give an analytic expression for the cost function, in the parameters p , q , and E .

A. First Approach

In this section, we will derive a closed form solution for the mean total cost of the zeroconf protocol.

We first introduce some notations. The probability matrix describing the DTMC depicted in Figure 1 is:

$$\mathbf{P} = (p_{ij})_{i,j=1,\dots,7} = \begin{pmatrix} \cdot & q & \cdot & \cdot & \cdot & \cdot & 1-q \\ 1-p & \cdot & p & \cdot & \cdot & \cdot & \cdot \\ 1-p & \cdot & \cdot & p & \cdot & \cdot & \cdot \\ 1-p & \cdot & \cdot & \cdot & p & \cdot & \cdot \\ 1-p & \cdot & \cdot & \cdot & \cdot & p & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \end{pmatrix}.$$

To emphasise the structure of the matrix, we mark zero entries as “.”. Please note that each row of a probability matrix corresponds to a state of the DTMC. In Table I, the mapping from state names to state numbers, as we will use in this paper, is given³. Therefore, if $\text{row}(s1) = i$ and

State	start	1st	2nd	...	nth	error	ok
$\text{row}(\cdot)$	1	2	3	...	$n+1$	$n+2$	$n+3$

TABLE I

MAPPING STATES TO ROW NUMBERS FOR n ARP PROBES

$\text{row}(s2) = j$ for states $s1$ and $s2$, then entry p_{ij} describes the probability to jump from state $s1$ to state $s2$. Since row is a 1-1 mapping, for our convenience, we will in the following refer to numbers only.

Matrix \mathbf{P} takes only probabilities into account. With the matrix \mathbf{C} we describe the costs of the transitions in the model. \mathbf{C} is defined as follows:

$$\mathbf{C} = (c_{ij})_{i,j=1,\dots,7} = \begin{pmatrix} \cdot & 2 & \cdot & \cdot & \cdot & \cdot & 8 \\ \cdot & \cdot & 2 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 2 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 2 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & E & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}.$$

Note that, if $p_{ij} = 0$, then also $c_{ij} = 0$. Furthermore, we generally must require that $c_{ii} = 0$, if i is an absorbing state, since otherwise the accumulated cost would not be finite.

We intend to calculate the mean cost that accumulates from system start until one of the two absorbing states is reached. This can be computed by means of a system of linear equations. Let $\underline{a} = (a_1, \dots, a_7)$ be a vector. We say that a_i describes the mean cost that accumulates, if state i is chosen to be the starting state. For the absorbing states $i = 6, 7$, we therefore know already that $a_i = 0$. The relation between the other total costs can be described by:

$$a_i = \sum_{j=1}^7 p_{ij}(c_{ij} + a_j), \quad (1)$$

for $i = 1, \dots, 5$. The meaning of this equation is as follows: $(c_{ij} + a_j)$ is the sum of the cost of transition $i \rightarrow j$

³Later, we will consider variants of the protocol where we allow n ARP probes to be sent, $n < 4$ or $n > 4$. Therefore we define $\text{row}(\cdot)$ more general.

and the mean accumulated cost of state j . So this is the total cost that incurs when the transition $i \rightarrow j$ is chosen. This cost is weighted with p_{ij} , since, outgoing from state i , the transition $i \rightarrow j$ is only taken with probability p_{ij} . We then have to sum over all possible target states $j \in \{1, \dots, 7\}$ of state i , to obtain the total cost of state i .

We can summarise all the equations of the form (1) in a single matrix-vector equation. Let $\mathbf{P}' = (p_{ij})_{i,j=1,\dots,5}$, the submatrix of \mathbf{P} that describes only the non-absorbing states $1, \dots, 5$. Let $\underline{w} = (w_1, \dots, w_5)$ be a vector, where $w_i = \sum_{j=1}^7 p_{ij}c_{ij}$. Let $\underline{a}' = (a_1, \dots, a_5)$ be the vector of the mean accumulated costs for states $1, \dots, 5$. Then the vector \underline{a}' is the solution to the matrix equation:

$$\underline{a}' = \mathbf{P}'\underline{a}' + \underline{w},$$

or better,

$$\underline{a}' = -(\mathbf{P}' - \mathbf{I})^{-1}\underline{w}.$$

($\mathbf{P}' - \mathbf{I}$ is regular, which is a conclusion from the Perron-Frobenius-Theorem for decomposable stochastic matrices [3]).

For our particular case, we are actually only interested in a_1 , the mean accumulated cost of state `start` (since it is the starting state). Since the DTMC is very small, we can derive a symbolic solution for a_1 , expressed in the variables p, q and E :

$$\mathcal{C}(p, q, E) := a_1 = \frac{8(1-q) + 2q(1+p+p^2+p^3) + qp^4E}{1-q+qp^4}. \quad (2)$$

B. Varying the Number of ARP Probes

The derivations so far were static in the sense that many assumptions made in the protocol draft [1] were directly incorporated into the model. In particular, we have assumed that a host always sends 4 ARP probes and that the waiting time between them is exactly 2 seconds. However, the number of probes n and the waiting time r are exactly the parameters that we wish to adjust such that the protocol runs with minimal cost. Therefore, we will generalise the model such that we can express the total cost of the protocol also depending on n and r .

Whereas the parameter r is easily incorporated in equation (2) (all that needs to be done is to replace the factor 2 by r), for n this is more difficult. The reason is that different choices of n do not only influence the numerical parameters of the model, but the *structure* of the model itself. However, the structural changes are straightforward. In Figure 1 we see the case for $n = 4$. If we consider now $n = 5$, all we have to do is to add another state and to connect it appropriately with the other states. In Figure 2, we can see the model for $n = 5$. We have introduced the state

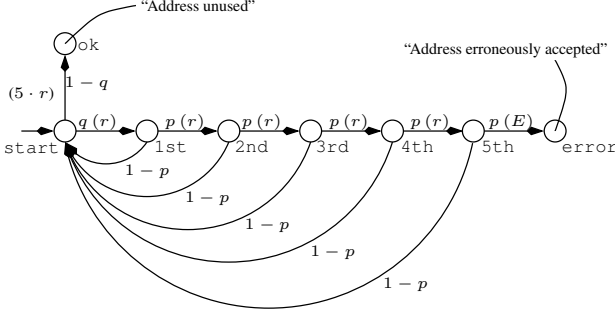


Fig. 2

MODEL THAT ALLOWS 5 ARP PROBES

5th, with one incoming transition with probability p and cost r from state 4th, an outgoing transition with probability $1-p$ and cost 0 to state start, and an outgoing transition to state error with probability p and cost E . If $n > 5$ ARP probes should be allowed, then states 6th, 7th, 8th etc. can be successively added to the DTMC, in the same fashion as for 5th before.

For each version of the DTMC, we can again derive a symbolic solution for a_1 , which depends now on r, p, q , and E . It can be shown that we can define the cost function as function of n . We obtain:

$$\mathcal{C}(n, r, p, q, E) = \frac{r \left(n(1-q) + q \sum_{i=0}^{n-1} p^i \right) + qp^n E}{1-q+qp^n}. \quad (3)$$

As is easily to verify, $\mathcal{C}(4, 2, p, q, E)$ is equal to $\mathcal{C}(p, q, E)$, as defined in Equation (2).

We can now formulate the optimisation problem as follows. For given p, q , and E , find the pairs $(n, r) \in \mathbb{N} \times \mathbb{R}^+$ such that $\mathcal{C}(n, r, p, q, E)$ is minimal. To get a feeling for the behaviour of the cost function, we first rewrite (3) a little bit. The sum in the numerator can be expressed in terms of the geometric series. We define

$$L_{p,q,r}(n) = r \left(n(1-q) + q \cdot \frac{1-p^n}{1-p} \right),$$

$R_{p,q,E}(n) = qp^n E$, and $D_{p,q}(n) = 1-q+qp^n$. Then

$$\mathcal{C}(n, r, p, q, E) = \frac{L_{p,q,r}(n) + R_{p,q,E}(n)}{D_{p,q}(n)}. \quad (4)$$

First, we observe that $D_{p,q}(n)$ is a factor that, although it depends on n , does not influence the behaviour of the function very much, at least, when we consider only small values of p (which is realistic). For sake of simplicity, we simplify the denominator to $1-q$, and since this is a constant

factor, it does not change the minima of the enumerator. Therefore, from now on we only consider $L_{p,q,r} + R_{p,q,E}$. We will consider the dependency of the cost function on n . The influence of r is considered in Section V.

In terms of n , we can see that there is indeed a trade-off: the left summand $L_{p,q,r}$ expresses the influence of the waiting time on the total cost. The right summand $R_{p,q,E}$ expresses the influence of E on the total cost. We can see that $L_{p,q,r}$ grows linearly with n , whereas $R_{p,q,E}$ falls off exponentially with n .

If we interpret L and R as functions $\mathbb{R} \rightarrow \mathbb{R}$, rather than $\mathbb{N} \rightarrow \mathbb{R}$, then both of them are continuous, and we obtain the derivative:

$$\frac{d(L_{p,q,r} + R_{p,q,E})}{dn} = r(1-q) + \left(E - \frac{r}{1-p} \right) qp^n \ln p.$$

Since $r/(1-p)$ is usually small compared to E (several orders of magnitude), we can ignore it and simplify the equation accordingly. Setting it then to zero, transforming it, and reinterpreting the result in the natural numbers again, yields:

$$n = \left\lceil \frac{\ln \left(\frac{r(q-1)}{qE \ln p} \right)}{\ln p} \right\rceil, \quad (5)$$

that expresses the maximal number of ARP probes that are allowed to be sent to achieve the minimal mean cost of the initialisation phase.

In the following, we show some sample graphs of the cost functions. We fix the values $E = 10^9$, $r = 2$, and $q = \frac{125}{8128}$ (which corresponds to the assumption that $m = 1000$ IP addresses on the local link are already in use). In Figure 3, we have plotted the continuous version of the cost function $\mathcal{C}(n, r, p, q, E)$. The x -axis ranges over n , the number of ARP probes. The y -axis shows the mean cost when n ARP probes are allowed to be sent. We show cost functions for different values of p , the probability that an ARP message gets lost. The values for p range from $p = 10^{-13}$ (the leftmost curve) to $p = 0.1$ (the rightmost curve), increasing by a factor of 100 from left to right. Note that the x -axis and the y -axis have not the same scale, which makes the linear asymptote appear steeper than it really is. The minima of the cost functions are more emphasised this way.

Figure 4 shows the optimal n , as expressed in Equation (5), depending on p . We see two curves. The solid curve depends on the lower x -axis, which ranges from $p = 0$ to $p = 0.5$. The result (left y -axis) is the maximal number of ARP probes such that the incurred mean cost is minimal.

The dashed curve highlights the range $p = 0$ to $p = 0.003$ (upper x -axis and right y -axis).

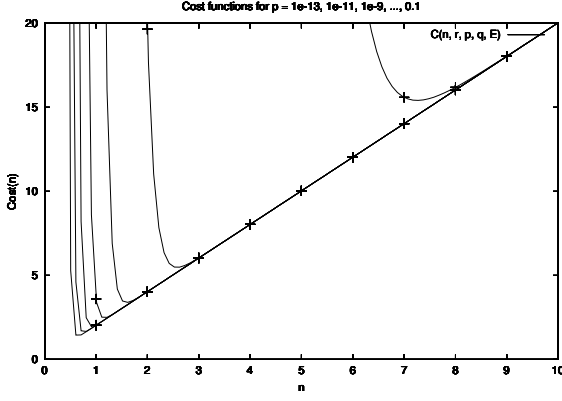


Fig. 3

COST FUNCTION FOR DIFFERENT VALUES OF p

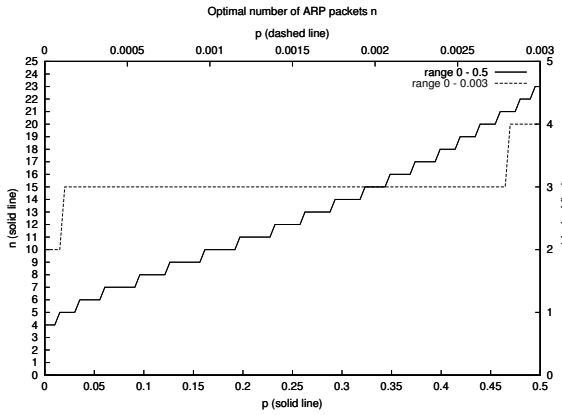


Fig. 4

NUMBER OF ARP PROBES TO INCUR MINIMAL COST

In [1], 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 $r = 2$. These assumptions are made explicitly for an ethernet network, although nothing about the speed (which can range over several order of magnitudes), size, and topology was said. Also no assumptions were 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 . We could now ask ourselves, what the value E must be such that the assumptions made in [1] become reasonable. We will choose very pessimistic parameters for p and q , and compute E for these values. We assume $p = 10^{-5}$ (which is enormously high for an ethernet), and $q = \frac{125}{8128}$, which again means that we assume circa 1000 hosts connected to the local link. We can com-

pute E from Equation (5):

$$E = \frac{r(q-1)}{qp^n \ln p}.$$

Using this formula with the chosen parameters we detect that E must lie roughly within the interval $[10^{16}, 10^{21}]$.

C. Absorption Probabilities

Apart from the incurred mean total cost, it is also interesting to know the probabilities to end up in state ok or state error (cf. Figure 2). Computing this is a standard problem for DTMCs, and the solution is described in detail in [2].

In the following we will give a concise derivation of the result. Although we will refer to the example that we have introduced in Section IV-A, the derivations hold in general.

The matrix P has the following structure:

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}' & \underline{p}_6 & \underline{p}_7 \\ \underline{0} & 1 & 0 \\ \underline{0} & 0 & 1 \end{pmatrix},$$

where \mathbf{P}' is the submatrix as defined in Section IV-A, and $\underline{p}_6, \underline{p}_7$ are column vectors. \mathbf{P} describes the 1-step-probabilities of a DTMC, i.e., p_{ij} is the probability to go to state j in one step, once state i has been entered. We know, that the power of \mathbf{P} , $(p_{ij}^{(k)}) = \mathbf{P}^k$, describes the k -step probabilities of the DTMC, i.e., $p_{ij}^{(k)}$ is the probability to go to state j in k steps, once state i has been entered. What are now the probabilities to end up in one particular absorbing state, say, 6, for example, given that we start in state 1? Apparently, this is $\lim_{k \rightarrow \infty} p_{1,6}^{(k)}$. How do we compute this?

We can write the powers of P as the following block matrix.

$$\mathbf{P}^k = \begin{pmatrix} \mathbf{P}'^k & \sum_{l=0}^{k-1} (\mathbf{P}')^l \underline{p}_6 & \sum_{l=0}^{k-1} (\mathbf{P}')^l \underline{p}_7 \\ \underline{0} & 1 & 0 \\ \underline{0} & 0 & 1 \end{pmatrix},$$

Therefore, since $\lim_{k \rightarrow \infty} (\mathbf{P}')^k = \mathbf{0}$,

$$\lim_{k \rightarrow \infty} \mathbf{P}^k = \begin{pmatrix} \mathbf{0} & (\mathbf{I} - \mathbf{P}')^{-1} \underline{p}_6 & (\mathbf{I} - \mathbf{P}')^{-1} \underline{p}_7 \\ \underline{0} & 1 & 0 \\ \underline{0} & 0 & 1 \end{pmatrix}.$$

Apparently, we have again to compute the inverse $(\mathbf{I} - \mathbf{P}')$, and to multiply it with \underline{p}_6 and \underline{p}_7 , respectively. The final results are then the first entry of the resulting vectors.

From the model, we can derive that the probability $\mathcal{E}(n, p, q)$ to reach state error is equal to

$$\mathcal{E}(n, p, q) = \frac{qp^n}{1 - q + qp^n}. \quad (6)$$

As we can see, the probability to accept an used address falls off exponentially with the number n of maximally allowed ARP probes to be sent, and approaches zero.

V. INTERDEPENDENCY OF n AND r

A. Optimising r

The model so far is not suitable to derive reasonable values for r , the waiting time, although it has a most important influence on the costs in the case that an unused address has been chosen.

There is a relation between r , the time to wait for an response, and p , the probability that no answer will be received. Intuitively one would assume that, the shorter r is, the higher p . We have no information about the exact relation between loss probability and waiting time. However, to demonstrate our approach, we define a probability function with an exponential fall-off with a rate λ as follows: $P(r, p, \lambda) = p + (1 - p)e^{-\lambda r}$. So $P(r, p, \lambda)$ is the probability that an answer is not received, depending on r . This probability approaches p for $r \rightarrow \infty$. So, $P(0, p, \lambda) = 1$ and $\lim_{r \rightarrow \infty} P(r, p, \lambda) = p$.⁴

We now consider the approximation (5) to compute n . Since P does not depend on n , we can consider it as a constant in the derivation of (5), so we can now write:

$$n(r, p, \lambda) = \left\lceil \frac{\ln \left(\frac{r(q-1)}{qE \ln P(r, p, \lambda)} \right)}{\ln P(r, p, \lambda)} \right\rceil. \quad (7)$$

This looks dangerous, since $P(r, p, \lambda)$ can be rather big, so that the disregard of the denominator of (4) might not be justifiable. But the effect we want to show is also visible with this approximation and would not vanish if we would use exact numbers. A typical plot for $n(r, p, \lambda)$ is given in Figure 5. What happens if we plug this function into the cost function, $\mathcal{C}(n, r, p, q, E)$? We define $\mathcal{C}_{new}(r, p, q, E, \lambda) = \mathcal{C}(n(r, p, \lambda), r, P(r, p, \lambda), q, E)$. \mathcal{C}_{new} is a function that heavily depends on r . In Figure 6, a plot of \mathcal{C}_{new} is shown. We see that the new cost function has a very peculiar form. Most importantly, we see that the function has several minima, which correspond to the steps of the function $n(r, p, \lambda)$. The reason for this is

⁴Choosing an exponential fall-off has some stochastic justification: since we do not know exactly how the fall-off does look like, we should assume as little as possible about it. An exponential fall-off is the natural choice then.

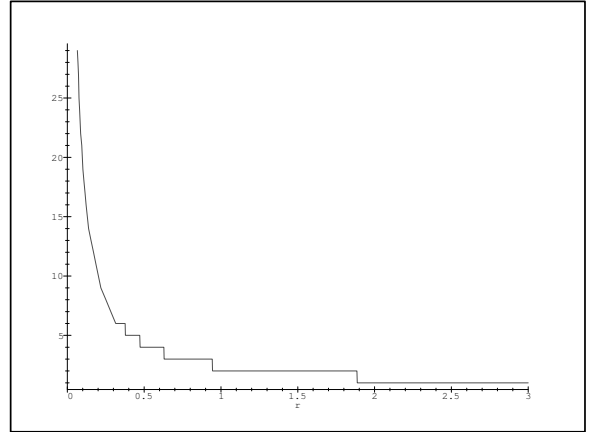


Fig. 5

$n(r, p, \lambda)$ FOR $p = 10^{-12}$, $\lambda = 10$, AND $r \in [0.1, 3]$

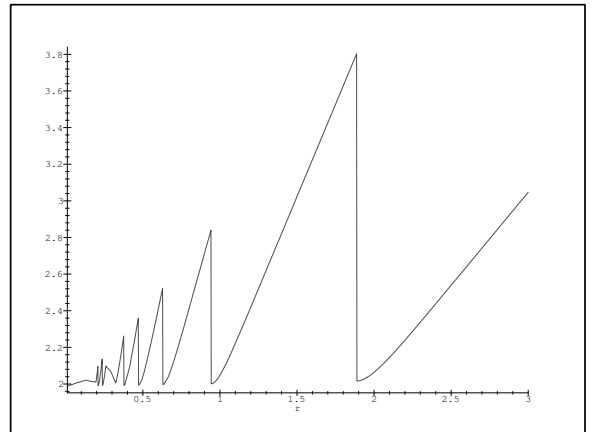


Fig. 6

$\mathcal{C}_{new}(r, p, q, E, \lambda)$ FOR $p = 10^{-12}$, $\lambda = 10$, $E = 10^9$ AND $r \in [0.1, 3]$

the following. The function $n(r, p, \lambda)$ is piece-wise constant. Let (a, b) be an interval where $n(r, p, \lambda) = c$. It is not surprising that \mathcal{C}_{new} is continuously growing in (a, b) , since the waiting time r is increasing and influences the overall cost. At the jump at b , $n(b+, p, \lambda) = c - 1$, i.e., the optimal number of ARP probes allowed to be sent is decremented by 1. This has a positive effect on the mean cost and results in a sharp drop at b . In fact, since $c - 1$ is the optimal number of ARP probes to be sent and b is the smallest value for which this optimum holds, the cost function will have a minimum at b , and *will be the same* at all jumps.

We see that we cannot give a clear answer to the ques-

tion which number of r is the optimum to achieve minimal cost. A small r would result in a high number of ARP probes sent, a large r would result in only a few ARP probes sent. Since sending a packet over the net results in costs that we are not accounting for in our model (overhead, increase in network traffic), it is probably a good policy to choose $n = \lim_{r \rightarrow \infty} n(r, p, \lambda)$ as the maximal number of ARP probes sent, and $r = \inf\{r \mid n(r, p, \lambda) = n\}$ as the waiting time. For our example, this means that $n = 1$, and $r \approx 1.9$.

B. Absorption probabilities, revisited

Since we assume that r influences the probably that an ARP probe remains unanswered, we must also reconsider the probability to reach state `error`, now in dependency of r . In Figure 7, we see the plot of the error function $\mathcal{E}_{new}(r, p, q, \lambda) = \mathcal{E}(n(r, p, \lambda), P(r, p, \lambda), q)$ for the same parameters for λ, p , and q as before. Please note that the y -axis has a logarithmic scale, *i.e.*, -18 stands for 10^{-18} , etc. We see that \mathcal{E}_{new} has discontinuities at the same points

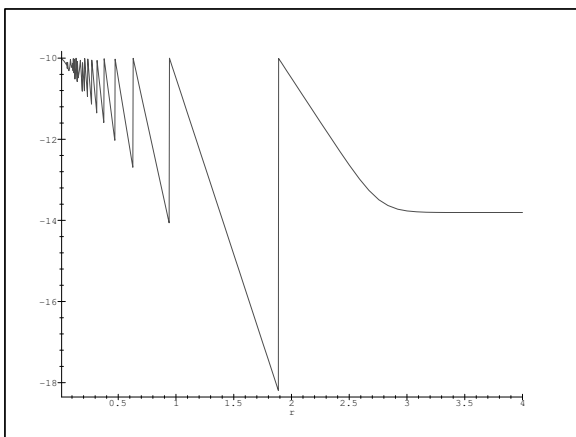


Fig. 7

PROBABILITY TO REACH STATE `ERROR`

for r as the functions $n(r, p, \lambda)$ and \mathcal{C}_{new} . Moreover, we see that the function is strictly *decreasing* in the continuous pieces of the function, and that there is a minimum: for r near to 1.9 (coming from the left), the probability to reach state `error` is approximately 10^{-18} , and for all other r , the probability is higher. For $r \rightarrow \infty$, the $n(r, p, \lambda)$ remains constant, and therefore, \mathcal{E}_{new} also approximates a constant.

The most important property of the three plots is: when the probability to go to the `error` state is minimal, then the mean cost is maximal! The other way round, when the mean cost is minimal, then the probability to reach state

`error` is maximal.

VI. DISCUSSION AND CONCLUSIONS

A. Adequacy of the Model

The model we have developed in this paper abstracts away from many details. The focus is on one host only. The rest of the system, *i.e.*, all the other hosts, and the network infrastructure, is reduced to numbers and functions. Moreover, the model is enhanced with costs, which describe the waiting time between the sending of ARP probes, and a not further specified penalty, if an IP address is erroneously adopted. The model depends on realistic values for the input parameters. In the following we describe which situation can be described by the model and which not, depending on the input parameters.

Loss Probability (P): We describe the loss probability of an ARP probe or its reply by means of a function P , which depends on the waiting time r . With this approach we capture the effect that a probe or reply gets lost depends also on the time that we wait for an reply. For the sake of simplicity, we have assumed in this paper that the loss probability falls off exponentially with a certain rate and approaches a certain threshold probability, p . To make the model more realistic, the probability function should be based on real world experience. Its characteristic depends on many different influences, very important ones being the underlying physical network and the speed of the network. The boundary loss probability can vary between 10^{-15} with a low variance for optical nets, and 0.4 with high variance for wireless nets, with a running microwave oven nearby. Taking all these influences into account to derive a probability function a priori is impossible. Realistic probability functions must be based on measures.

Number of hosts connected to the network (m): The number of hosts that we assume to be on the local link determines the collision probability, q . In a normal situation we can assume that m is not changing very quickly, so that it is reasonable to assume it constant during a initialisation phase. However, there are situations imaginable where this is not the case. We can imagine a local network with m hosts connected to it, and that they all try to configure themselves at the same time. This kind of situation can occur when the power comes back after a power failure, which knocked out all the hosts. The configuration attempts of the hosts happen at nearly the same time, triggered by the power coming back. This causes a burst of ARP probes sent around, and a sharp increase in the number of m . Such a situation is not covered by our model.

Penalty for erroneously accepted IP address (E): A special parameter is E . E represents the cost that incurs when

an used address is erroneously adopted by a host. E describes many different aspects of this error. It subsumes all the bad effects that are likely to occur when two hosts have the same address. From a practical point of view, the situation will be detected by both hosts, and one or both then reconfigure themselves, which takes some time. So perhaps we must estimate the mean time which is needed to “clean up” the situation. However, there are other aspects which can not be easily expressed in seconds. There is the question if there happens any damage if a connection is severed due to reconfiguration. Reconfiguration takes its time, but the reestablishment of connections might also take its time. Moreover, perhaps it is a TV set which communicates via IP with a VCR. A viewer might see it as absolutely unacceptable when the screen blacks-out for more than $n \cdot r$ seconds, because it must reconfigure its IP address. Even more since this happens of course always at the most thrilling moment of the movie.

Apparently, we have also psychological aspects to take into account, which can influence the success of a product (A product that reconfigures too often does no sell). We see that a technical problem can reach into dimensions which are usually not accounted for in a standard evaluation of a protocol. Assigning numbers to these aspects is certainly difficult and can not be done with exact methods.

B. Conclusions

In Section I, we have posed several questions about the protocol that we wanted to answer by means of our model. We will now comment on them.

- “Is it actually needed to send *four* probes?” Our model allows to give an answer on this question. We have seen in Section V-A that there are more than one pair of n and r which minimise the expected cost of the protocol. Hence, we can give an answer to this question: either the modified cost function C_{new} has a cost minimum where n is smaller than 4, or not.
- “Are there variations of the protocol which are equivalent except that configuration goes faster?” The only variation of the protocol we allow in the model are the input parameters, basically, n and r . In this respect we can say, yes, there are variations of the protocol which are faster, if we choose a value pair (n, r) that yields minimal cost.
- “What is the probability that an address collision occurs in the initialisation phase?” With the methods described in Section IV-C and Section V-B we can derive the probabilities that an address collision occurs. Moreover, we have also gained insight in the interdependence of cost and collision probability: they counteract.

Although the model is very abstract, we can learn something from it. We have seen that the parameters p , P , q and

E are not sufficient to determine an absolute minimum of the cost function in the number of n and r . To actually decide about a value, other factors have to be taken into account, for example, the overhead caused by sending an ARP probe, or the overall network traffic.

Moreover, we have seen that the probability for an address collision is at its highest, if the parameters are chosen for optimal cost. The other way round, if we chose n and r such that the collision probability is at its smallest value, then the mean cost is maximal. We can clearly see that reliability has its cost.

Acknowledgements We thank Peter van der Stok from Philips Research, Eindhoven, for valuable suggestions. This work is supported by PROGRESS project TES4999: Verification of Hard and Softly Timed Systems (HaaST).

REFERENCES

- [1] Stuart Cheshire and Bernard Adoba. Dynamic configuration of IPv4 link-local addresses. <http://www.ietf.org/internet-drafts/draft-ietf-zeroconf-ipv4-linklocal-04.txt>, July 2001. DRAFT.
- [2] Vidyadhar G. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, London, Glasgow, Weinheim, 1995.
- [3] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.