UNIVERSITÄT KOBLENZ · LANDAU
Institut für Informatik

FB 4 Informatik

Avoidance of Routing Loops

Frank Bohdanowicz
Harald Dickel
Christoph Steigner

Nr. 1/2009

Arbeitsberichte aus dem Fachbereich Informatik

Arbeitsberichte des Fachbereichs Informatik

ISSN (Print): 1864-0346
ISSN (Online): 1864-0850

Herausgeber / Edited by:

Der Dekan:
Prof. Dr. Zöbel

Kontaktdaten der Verfasser

Frank Bohdanovicz, Harald Dickel, Christoph Steigner
Institut für Informatik
Fachbereich Informatik
Universität Koblenz-Landau
Universitätsstraße 1
D-56070 Koblenz
EMail: bohdan@uni-koblenz.de, dickel@uni-koblenz.de, steigner@uni-koblenz.de

Avoidance of Routing Loops

F. Bohdanowicz, H. Dickel, Ch. Steigner
Email: {bohdan, dickel, steigner}@uni-koblenz.de
Institute for Computer Science, University of Koblenz-Landau

***Abstract*—We introduce a new routing algorithm which can detect routing loops by evaluating routing updates more thoroughly. Our new algorithm is called Routing with Metric based Topology Investigation (RMTI), which is based on the simple Routing Information Protocol (RIP) and is compatible to all RIP versions.**

In case of a link failure, a network can reorganize itself if there are redundant links available. Redundant links are only available in a network system like the internet if the topology contains loops. Therefore, it is necessary to recognize and to prevent routing loops. A routing loop can be seen as a circular trace of a routing update information which returns to the same router, either directly from the neighbor router or via a loop topology. Routing loops could consume a large amount of network bandwidth and could impact the end-to-end performance of the network. Our RMTI approach is capable to improve the efficiency of Distance Vector Routing.

I. Introduction

Fault tolerant internet topologies must contain loops, but loops are a major problem for all routing algorithms. Loops in networks provide alternative paths between nodes in case of a link loss, but loops also complicate the detection of correct and optimal paths between the nodes. Most of the communication and computation work in current routing algorithms is done only in order to cope with loops. Routing loops occur within an Autonomous System (AS) as a result of temporary inconsistency arising during the convergence process.

The simple Routing Information Protocol (RIP) [8] cannot adequately cope with routing loops. An approach like *split horizon* fails if the network topology contains loops. The routing process is severely damaged because out of date reachability information may be proliferated by routing updates along the topology loop again and cause the *counting to infinity* (CTI) problem [8]. The CTI problem is well known as an unsolved drawback of RIP (Split Horizon Hack [19]). The only way that RIP can achieve some kind of convergence with CTI situations is to limit the counting metric to 16. But even this is often not possible, because a lot of misguided data traffic may congest the links on which the *counting to infinity* takes place and thus the state of the routing tables cannot converge. Approaches like Open Shortest Path First Protocol (OSPF) do not solve the problem of routing loops entirely, because of the occurrence of transient routing loops [4], [6], [20]. Transient loops occur as part of the normal operation of the OSPF protocol due to the different delays in the propagation of routing updates to different parts of the network.

In order to overcome the deficiencies of RIP, we developed the *Routing with Metric based Topology Investigation (RMTI) Protocol* [16] that is a crucial extension to the local processes of the Routing Information Protocol (RIP) and is entirely downward compatible to RIP. However, RMTI can prevent CTIs by observing the metric information advertised by routing updates in order to recognize which updates have made their way along a loop.

Further on we will demonstrate that routing loops can be detected by recognizing *Simple Loops* and *Source Loops*. A Simple Loop (figure 1a) is a path which leaves a distinct router at one interface and comes back to the same router on another interface, *without having passed* through the same router on other interfaces. Thus, a Simple Loop describes a loop in the topology. A Source Loop (figure 1b) is a path which leaves a distinct router at one interface and comes back to the same router on another interface, *having passed* through the same router on other interfaces.

We are able to recognize these loops by not strictly deleting old routing update information as soon as new and better updates appear; instead, we use a careful selection of this information in order to detect Simple Loops and Source Loops. We perform this kind of loop recognition by presenting two equations to be applied on stored routing metrics. These equations also provide us with decision criteria to reject malicious routing updates which could cause routing loops and the *counting to infinity* problem.

Our new RMTI algorithm uses the entire topology information about loops within the network. This enables RMTI to decide on the acceptance or rejection of all updates which provide alternative routes more carefully. To make decisions of this kind we have provided three operation modes. The *Normal Mode* rejects only alternative routes which have been verified as invalid. The *Strict Mode* accepts only the routes which have been verified as valid. The *Careful Mode* accepts valid routes and marks routes which seem to be invalid for a further evaluation step.

The Normal Mode prevents all the invalid decisions of the classical RIP algorithm, and improves the stability and convergence properties of the routing algorithm. The Strict Mode prevents the occurrence of CTIs entirely, thus, it improves the stability of the routing algorithm. The Careful Mode combines the features of the Normal Mode and Strict Mode by accepting doubtful updates in contrast to the Normal Mode but still safely preventing CTIs. To find out which of the modes is preferred depends on the requirements which are put onto the networks. The Normal Mode is an improvement in comparison to the RIP algorithm, but it cannot prevent CTIs entirely.

If the occurrence of CTIs is not acceptable, the Strict or Careful Mode are the preferred options. The Strict Mode

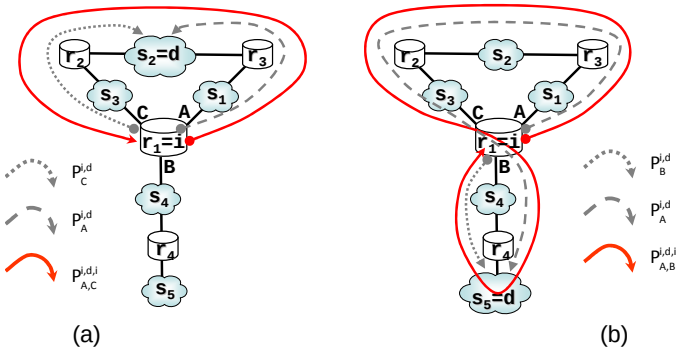

Fig. 1. Path combinations. (a) shows the Simple Loop $P_{A,C}^{i,d,i}$ with $d = s_2$ as a combination of the two paths $P_A^{i,d}$ and $P_C^{i,d}$. (b) shows the Source Loop $P_{A,B}^{i,d,i}$ with $d = s_5$ as a combination of the two paths $P_A^{i,d}$ and $P_B^{i,d}$.

results in a less number of changes in the forwarding tables and favors the network stability. In contrast to this, the Careful Mode provides better convergence properties. As an additional criterion for the selection of suitable RMTI modes, the local topology information about the router can be used. The more interfaces a router has and the more loops that are recognized, the greater the probability of CTIs occurring. The number of interfaces and loops of a router can, thus, automatically be used to find the best RMTI mode.

We have done comprehensive tests with different network topologies in order to analyze the RMTI behavior. The remainder of this paper is organized as follows. In Section II we present the principles of our new RMTI approach. In Section III we present our protocol together with some characteristic examples of routing loop detection and update rejection. In Section IV we depict our implementation and in Section V we review related works. We close with Section VI where we present our conclusions.

II. Design Rationale

The basic idea of RMTI is to discover all local loops which start and end at a particular router. This knowledge is used to prevent routing loops and *counting to infinity* situations. In the following, we explain some basic terms and notations used in our network model. A network consists of routers and subnets. Routers and subnets are connected via interfaces, so a network is represented by a network graph where the nodes of the graph are the routers and the edges are the subnets. Each connection between a node and an edge in this network graph is an interface and is marked with an unique interface identifier (figure 1).

Therefore we have a set of routers $R = \{r_1, \ldots, r_k\}$, a set of subnets $SN = \{s_1, \ldots, s_n\}$, and a set of interfaces $IF = \{\text{if}_1, \ldots, \text{if}_m\} = \{A, B, C, \ldots\}$. A router $r \in R$ is identified by its interfaces, moreover r is defined by the set of its interfaces $r = \{A, B, C, \ldots\}, r \subseteq IF$ and $\forall A \in IF, A \in r_i \Rightarrow A \notin r_j$ for $r_i, r_j \in R, i \neq j$. That is, an interface is an element of one unique router only.

Interfaces are connected to subnets and the topology of a network graph is given by the relation $CON \subseteq IF \times SN$ where $(\text{if}_i, s_j) \in CON$,$\text{if}_i \in IF, s_j \in SN$, if and only if interface if_i is connected to subnet s_j.

A path through the network graph is a sequence of hops where a hop is an elementary step from a router $i \in R$ via outgoing interface $O \in i$ to an adjacent router j via incoming interface $I \in j$ using subnet $s \in SN$. So a hop is defined by a 3-tuple $H^{i,j} = (O, s, I)$, whereas $(O, s), (I, s) \in CON$ and a path $P^{i,j}$ from router i to router j is given by its hops $P^{i,j} = (H_1, H_2, \ldots, H_l)$. The metric of $P^{i,j}$ is $m^{i,j} = l$ which is simply the number of hops. This kind of counting the metric corresponds to the hop-count metric used by RIP.

A. Simple Loops and Source Loops

Of special interest are closed paths that connect a subnet d via two different interfaces of the same router i (figure 1). They form a loop within the network consisting of subnet d and it is tempting to assume that such a loop offers two distinct useful paths to subnet d. This however, has to be considered very carefully.

Figure 1a shows the network graph of an example network topology. We examine paths from router $r_1 = i$ to the destination subnet $d = s_2$. There is a path P_A^{i,s_2} via interface A and a path P_C^{i,s_2} via interface C from router i to network s_2. If we combine these paths, we obtain a closed path $P_{A,C}^{i,d,i}$, which starts at router i via interface A and ends at router i via interface C.

In figure 1b the position of the destination subnet d is shifted to s_5. Now there is a path P_B^{i,s_5} from router i via interface B to network $d = s_5$ and a path P_A^{i,s_5} from router i via interface A to network d. If we combine these paths, we obtain a closed path $P_{A,B}^{i,s_5,i}$ once again. Both paths build a loop. But there is a major difference between the path $P_{A,B}^{i,s_5,i}$ and the path $P_{A,C}^{i,s_2,i}$. Path $P_{A,B}^{i,s_5,i}$ traverses router i in between, whereas path $P_{A,C}^{i,s_2,i}$ does not.

In the topology of figure 1 there is a topology loop between the interfaces A and C at router i. If the path P_C^{i,s_2} is no longer usable (due to a link or router failure on that path), there is an alternative path P_A^{i,s_2} to subnet s_2 which router i might use.

But there is no loop in the topology between the interfaces A and B at router i. If the path P_B^{i,s_5} is no longer usable, there is no alternative path P_A^{i,s_5} to subnet s_5 which router i might use since P_B^{i,s_5} is part of the path P_A^{i,s_5}. Therefore we have to distinguish between these two different types of closed paths. First, we define the term Simple Loop, which is a path from router i, traversing a subnet d, ending at router i, but never passing router i in between.

Definition 1: A Simple Loop is a path $P_{A,B}^{i,d,i}$ where $A, B \in i, O_1 = A,\ I_l = B, \exists n\ 1 \leq n \leq l\ s_n = d$, and $\forall I_j\ 1 \leq j < l\ I_j \notin i$.

We denote the set of all Simple Loops within a network with SIL and the metric of a Simple Loop $P_{A,B}^{i,d,i} \in SIL$ is $silm_{A,B}^{i,d,i} = m_{A,B}^{i,d,i} = l$.

Second, we define the term Source Loop, which is also a path from router i, traversing a subnet d, ending at router i, but this time passing router i in between.

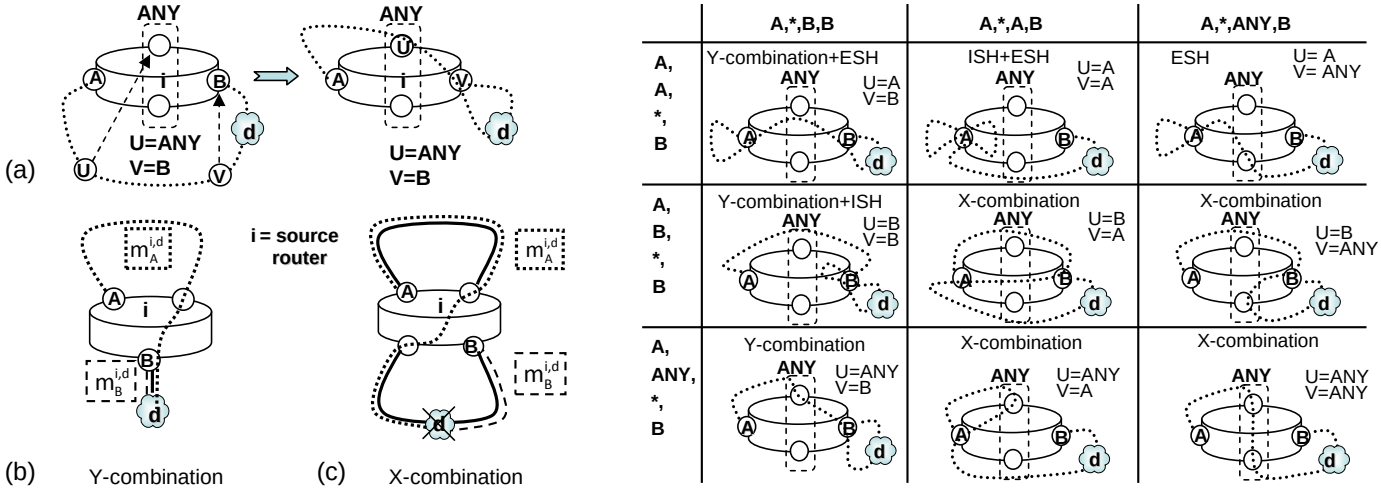

Fig. 2. Source Loop combinations. (a) shows a simple combinatorial scheme to create all Source Loop combinations shown by the table to the right. The split horizon rule avoids the four Source Loop combinations marked with ISH or ESH. Internal split horizon (ISH) means that the split horizon rule takes place at the local router i. External split horizon (ESH) means that the split horizon rule takes place at the neighbor router. (b) shows the Y-combination $P_{A,B}^{i,d,i} \in SOL$ with $U =$ANY, $V = B$ and (c) shows the X-combination $P_{A,B}^{i,d,i} \in SOL$ with $U =$ANY, $V =$ANY. In each case the metric of the Source Loop is $m_{A,B}^{i,d,i} = m_A^{i,d} + m_B^{i,d} - 1$.

Definition 2: A Source Loop is a path $P_{A,B}^{i,d,i}$ where $A, B \in i, O_1 = A$, $I_l = B$ $\exists n\ 1 \leq n \leq l$ $s_n = d$, and $\exists I_j\ 1 \leq j < l$ $I_j \in i$

We denote the set of all Source Loops within a network with SOL and the metric of a Source Loop $P_{A,B}^{i,d,i} \in SOL$ is $solm_{A,B}^{i,d,i} = m_{A,B}^{i,d,i} = l$.

In a network topology only Simple Loops provide an usable alternative path to a destination subnet. To gather information on the existence of Simple Loops at a router, we have to exclude the possibility that a closed path $P_{A,B}^{i,d,i}$ is a Source Loop (like in figure 1b).

B. The X- and the Y-test

In order to recognize all appearances of Source Loops, we introduce a combinatorial scheme which captures all Source Loops (figure 2). Every variation of a Source Loop can be formed by a simple scheme. A router may have an interface (A) for the start of a Source Loop and an interface (B) for the end of a Source Loop. The router may have further interfaces which we subsume into a group of interfaces which are denominated ANY. If we connect the interfaces A and B via a loop path we get a Simple Loop. If, however, we affix two interface representatives U and V in the loop and place the two representatives either on the interfaces A,B or ANY just by moving U and V to the interfaces (A, B, ANY), we get all together $3^2 = 9$ combinations of the Source Loops. Figure 2 shows all 9 Source Loop variations. Furthermore, we distinguish two basic types of Source Loops, the X-combination and the Y-combination. The X-combination is a connection of two different Simple Loops, whereas the Y-combination is formed by connecting a Simple Loop with a path $P_{B,B}^{i,d,i}$, which starts and ends via interface B where $m_B^{i,d}$ is the lowest possible metric from router i subnet d.[1]

In order to detect Source Loops, we need to identify the Simple Loop with the lowest metric out of a set of recognized Simple Loops of different metric size. This can be done by simply inspecting all Simple Loop metrics.

The minimal Simple Loop metric (msilm) between two interfaces A and B on router i is:

$$\text{msilm}_{A,B}^{i} = \min\{\text{silm}_{A,B}^{i,d,i} \text{ for all subnets } d\}$$

Further on we define the minimal return path metric (mrpm) which can be derived from the minimal Simple Loop metric (msilm). The minimal return path metric (mrpm) via an interface A on router i is:

$$\text{mrpm}_{A}^{i} = \min\{\text{msilm}_{A,B}^{i} \text{ for all interfaces } B \neq A \text{ of router } i\}$$

Based on the minimal return path metrics, we are able to give two equations which are sufficient to rule out that a path is a X- or a Y-combination.

A path $P_{A,B}^{i,d,i}$ with metric $m_{A,B}^{i,d,i}$ is no X-combination, if the following equation holds:

$$m_A^{i,d} + m_B^{i,d} - 1 < \text{mrpm}_A^i + \text{mrpm}_B^i \qquad (1)$$

If a router i is aware of two distinct paths $P_A^{i,d}$ and $P_B^{i,d}$ to a destination subnet d, then equation 1 offers a test to rule out that the combined path $P_{A,B}^{i,d,i}$ is an X-combination. If equation 1 holds, then the metric of the combined path $m_{A,B}^{i,d,i} = m_A^{i,d} + m_B^{i,d} - 1$ is less than the sum of the minimal return path metrics $\text{mrpm}_A^i + \text{mrpm}_B^i$. Therefore the path $P_{A,B}^{i,d,i}$ is *too short* to be an X-combination (figure 2c). This test is called the *X-test*.

[1]Therefore this path is used by router i to reach subnet d when the routing process is converged.

A path $P_{A,B}^{i,d,i}$ with metric $m_{A,B}^{i,d,i}$ is no Y-combination, if the following equation holds:

$$m_A^{i,d} < \text{mrpm}_A^i + m_B^{i,d} \quad (2)$$

To form a Y-combination the lower boundary of $m_A^{i,d}$ is the sum of the minimal return path metric mrpm_A^i and the metric $m_B^{i,d}$ to reach destination subnet d via interface B. If equation 2 holds, the path $P_{A,B}^{i,d,i}$ is *too short* to be a Y-combination (figure 2b). This test is called the *Y-test*.

C. Detection of Source Loops in Distance Vector Routing

So far we analyzed the properties of paths in a network graph. We developed tests to infer from the metric of a closed path in the network graph to eliminate any possibility that this path is a Source Loop and therefore must be a Simple Loop.

To apply these considerations to distance vector routing, we now have to take into account the trace of routing update information of the distance vector routing process. The metrics to destination subnets are sent via update messages between adjacent routers throughout the network. So we have to look at the potential succession patterns of update messages and the chronological sequence in which these messages arrive at a router. Furthermore we have to consider that a computer network may change its topology. As routers and links are added or removed, the underlying network graph is changes. It is possible that the metrics of update messages are calculated by obsolete data and therefore are out-dated. To cover all these situations, a router has to store some information about the network history.

Assume an update message arrives at router i via interface A containing the metric $m_A^{i,d}$ to subnet d 1. If router i already has an entry in his routing table to subnet d via an interface $B \neq A$ and metric $m_B^{i,d}$, we can build a closed path $P_{A,B}^{i,d,i}$ with metric $m_{A,B}^{i,d,i} = m_A^{i,d} + m_B^{i,d} - 1$. If this path $P_{A,B}^{i,d,i}$ is a Simple Loop then there is a loop in the network graph between the interfaces A and B of router i. But it is also possible that $P_{A,B}^{i,d,i}$ is a Source Loop. In this case router i must have sent out an update message via an interface U containing metric $m_V^{i,d}$ sometime in the past and this information has traveled through the network in a loop, finally returning to router i via interface A.

The table in figure 2 shows all Source Loop combinations. We assume that the split horizon rule is in operation. If U and V mark the same interface, then the split horizon rule of the local router (internal split horizon) eliminates the possibility, that such a Source Loop combination occurs. If U and A mark the same interface (first row of the table), then the split horizon rule of the neighbor router behind interface A (external split horizon) eliminates the possibility that such a Source Loop combination occurs. This rules out four of the nine entries in the table shown in figure 2.

To rule out that $P_{A,B}^{i,d,i}$ is a Y-combination like figure 2(b), we then perform the Y-test. But we should be cautious at this point. The Y-test is done using the metric $m_B^{i,d}$, which is usually the metric of the actual entry to subnet d in the routing table of router i. But there is the possibility that this metric has changed in the near past and no longer reflects the correct contribution of $m_B^{i,d}$ within the path $P_{A,B}^{i,d,i}$. In particular, if $m_B^{i,d}$ has increased in the span of time the update messages needed to travel through the network, building a Source Loop returning via interface A, then the Y-test may fail. To avoid this trap, we have to perform the Y-test using the lowest metric $m^{i,d}$ together with the associated interface the router i has recognized in the past.

This Y-test can also be used as a replacement for the X-test. To do this, let us take a look on the X-combination shown by figure 2(c). Such an X-combination may arise if the router i changes the interface according to its entry to subnet d before the Source Loop is closed by the arrival of the update message containing the metric $m_A^{i,d}$ via interface A. This metric $m_A^{i,d}$ fails the X-test and therefore the path $P_{A,B}^{i,d,i}$ is not accepted as a Simple Loop. But we can use the Y-test to get the same result. Figure 3 shows a concrete example of the occurrence of a X-combination and the application of the X- and Y-tests in this case. Within this topology subnet d is reachable from router i via interface D with metric 2 and via interface B with metric 3. Router i selects the path $P_D^{i,d}$ with the lower metric, and propagates corresponding update messages via its interfaces. This information may circulate through the network loop between interface C and A. Assume subnet d becomes inaccessible. Now the following three events cause an X-combination (see figure 3):

1) Router i receives an update via interface D that subnet d is unreachable while an update via interface B still announces subnet d with metric 3. Therefore i switches from interface D to interface B to reach subnet d.
2) Router i receives an update via interface B that subnet d is unreachable.
3) Router i receives an update via interface A that subnet d is reachable with metric 5.

The X-test $m_A^{i,d} + m_B^{i,d} - 1 \not< \text{mrpm}_A^i + \text{mrpm}_B^i$, $5+3-1 \not< 3+4$ detects the X-combination and therefore router i should reject the update which offered the path $P_A^{i,d}$. The Y-test performed

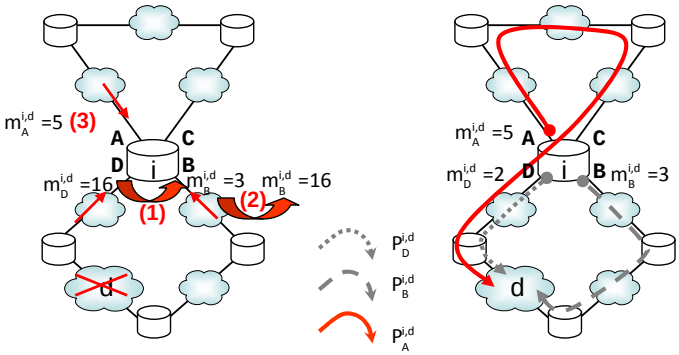

Fig. 3. Occurrence of an X-combination. (1) Router i receives an update via interface D that subnet d is unreachable and switches from interface D to interface B to reach subnet d. (2) Router i receives an update via interface B that subnet d is unreachable. (3) Router i receives an update via interface A that subnet d is reachable with metric 5 (source Loop).

with the latest valid metric $m_B^{i,d}$ to subnet d *fails* to detect a Y-combination $P_{A,B}^{i,d,i}$, since $m_A^{i,d} < \text{mrpm}_A^i + m_B^{i,d}$, $5 < 3+3$. But the Y-test performed with the lowest metric to subnet d router i has used in the past, which is $m_D^{i,d}$ in this case, detects the Y-combination $P_{A,D}^{i,d,i}$, since $m_A^{i,d} \not< \text{mrpm}_A^i + m_D^{i,d}$, $5 \not< 3+2$ and the Y-test *fails not*.

As stated above, in case the metric to a subnet has increased, the old metric has to be stored together with a change of the interface by router i in order to perform the Y-test. If we always use the lowest metric to a subnet together with the corresponding interface stored at a router, the X-test is superseded by the Y-test. Therefore, it is sufficient to perform the Y-test to detect all the possible Source Loop combinations shown by figure 2.

III. RMTI Paradigms

A RMTI router gathers knowledge about Simple Loops and Source Loops in its local environment by applying the Y-test to all incoming routing updates of one destination subnet, which, however, arrive on different links of the same router. As described in Section III, the functionality of the X-test can be completely replaced by the Y-test if it is not used with the very last metric of the affected route but with the lowest metric of the route within the recent update periods. The knowledge of the RMTI-algorithm is represented by two basic parameters which are used for the decision whether to accept or reject a routing update.

The *first* basic parameter is the *minimal Simple Loop metric (msilm)*, as described in Section III. This is the metric of a loop in the topology which starts and ends on two distinct interfaces of the same router. All msilm metrics are stored in a local table. A Simple Loop entry in the msilm table consists of the two related interfaces and the minimal loop metric. The msilm entry is calculated from the path combination of two alternative routes to one destination subnet and is verified with the Y-test. If an interface has no Simple Loop with another interface of the router, the msilm entry for this pair of interfaces is set to RMTI-infinity which is 31 by default. RMTI-infinity is calculated from the path-combination with metric infinity for both: $16 + 16 - 1 = 31$. The msilm table remains rather small because the maximum number of Simple Loops depends on the number of the router's interfaces. It does not increase as long as the number of Simple Loops does not increase in the local environment of the router. The general computation complexity of the search and insert operations is dependent on the number of routes to check, and therefore on the size of the network. As the msilm table is symmetric, it can be linearized in order to achieve effective storage.

The *second* basic and more important parameter for the Y-test is the *minimal return path metric (mrpm)*. The mrpm entry is the smallest msilm entry of a specific interface, which means it is the smallest metric of a loop which starts at a distinct router interface and returns to another arbitrary interface of the same router. The mrpm entry has to be recorded together with the interface within the mrpm table. If an interface has no loop with any other interface of the router the mrpm entry is set to RMTI-infinity. As the number of interfaces of a router is limited, the number of table entries is limited too and is not dependent on the network size.

The RMTI decisions based on these parameters can be optimized by three different operation modes:

1) The *Normal Mode* rejects only alternative routes which have been verified as invalid.
2) The *Strict Mode* accepts only the alternative routes which have been verified as valid.
3) The *Careful Mode* accepts only the alternative routes which have been verified as valid and marks the invalid for further consideration.

All kinds of routing loop situations are assigned to one of the three operation modes. Depending on the detected topology and the possible routing loops, our RMTI-algorithm can choose the required operation mode to prevent all CTI situations without increasing the convergence time after a link failure. Thus, the implementation of our RMTI-algorithm operates as one autonomous mode.

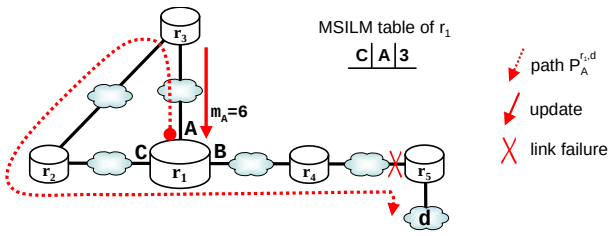

Fig. 4. Assuming the link between routers r4 and r5 becomes corrupted. The router r1 will propagate this unreachability update to its neighbors r2 and r3. But in this span of time r2 sends an update with old information to r3, which then is also announced back to r1. In this situation r1 must verify the correctness. Because there is no entry in the msilm table for interfaces A and B, the update will be rejected.

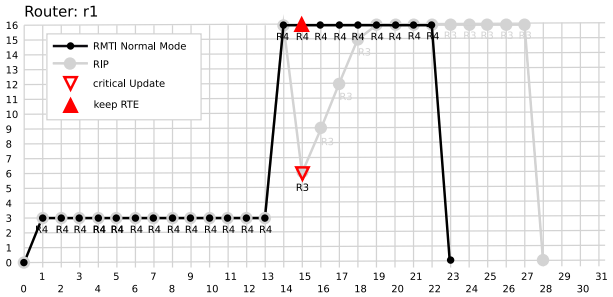

Fig. 5. The curve diagram shows the test results of the RMTI Normal Mode in comparison to standard RIP captured on router r1. The y-axis maps the metrics and the x-axis maps the number of incoming routing updates. Both curves follow the metric of the route to subnet d. The notation below the curves' points represents the router by which r1 learned the route to subnet d. The triangles describe the rejection of a critical update and for keeping the corresponding routing table entry (RTE). They indicate that the RMTI Normal Mode (black curve) on r1 receives and rejects the critical update that would have started counting to infinity. In the background (gray curve) the progression of counting to infinity is drawn when standard RIP is used. The advance of the Normal Mode would be even greater if the metric used by RIP would not be limited to 16 (=infinity).

A. Normal Mode

The Normal Mode was designed in order to recognize and to prevent the *counting to infinity* problem within a network of single loops. An interface is part of a multiple loop if it has more than one entry in the msilm table. It is part of a single loop if it has exact one entry in the msilm table. A Source Loop is identified by testing the metric of a newly incoming routing update.

The following example describes the prevention of CTI situations within the topology described by figure 4: Assuming that the link to network d via interface B becomes corrupted, this is indicated by an update which router r1 receives via this interface B, and therefore the former entry becomes invalid and is marked with metric infinity. Next r1 receives a routing update to subnet d via interface A $\neq$ B. At this point RMTI switches to CTI avoiding. The decision to accept or to reject this update is based on the information about Simple Loops that RMTI has collected during loop detection. Because the msilm table holds no entry, which substantiates the existence of a Simple Loop between interface A and B, the update will be rejected (figure 5).

The Normal Mode is based on two operational steps. Step one is applied if a valid route to a network is given and another update appears on the same router on a different interface. Then the minimal Simple Loops in the local environment of the router can be detected by applying the Y-test. Step two is applied if a route is corrupted and an alternative route has to be accepted. In this situation a decision can be made by inspecting all the gathered Simple Loops to see whether an alternative route is available. This is done by a simple rule:

If there exists a Simple Loop between the new route's interface and the old route's interface, then an alternative route may exist. If no Simple Loop exists, then the new route must be a Source Loop and has to be rejected.

Thus, in the Normal Mode a distinction is made between finding Simple Loops and avoiding *counting to infinity*.

Complexity: In the Normal Mode only one test is performed. It compares the path combination to the corresponding msilm entry. If there is no entry, no Simple Loop was recognized between the accordant interfaces. In this case the highest possible value (RMTI-infinity) is used, where every path combination of two valid routes is always less. In the Normal Mode, the mrpm and msilm tables have to be accessed.

The Normal Mode is a simple CTI avoiding mechanism, but only for more or less simple topologies. However, if we now extend the topology above by one additional loop (a Simple Loop between interfaces A and B of r1), then the Normal Mode is not sufficient anymore (figure 6). Now the msilm table has more than one entry for specific interfaces, which indicates multiple loops. The Normal Mode would prevent all CTI situations on router r4. As all interfaces on routers r1, r2 and r3 are part of a Simple Loop, the Normal Mode rule is not sufficient any more, since an alternative path is available from each interface. To solve this problem, we introduce the Strict Mode, which tests updates by a different rule.

B. Strict Mode

The following example describes the prevention of *counting to infinity* situations within the multiple loop topology described by figure 6: Assume that the link between the routers r4 and r5 becomes corrupted. The router r4 will propagate the unreachability of subnet d to its neighbors r1 and r3. Just before r1 sends an update to router r2, r2 sends an update with old routing information (d reachable via r1) to r3, which

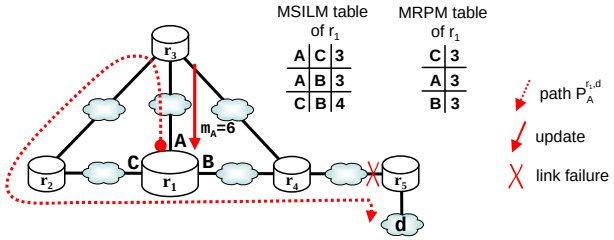

Fig. 6. One additional loop is added between r4 and r3. This leads to a new situation for r1, because all its interfaces are now part of Simple Loops. Every interface of r1 has a mrpm entry of 3.

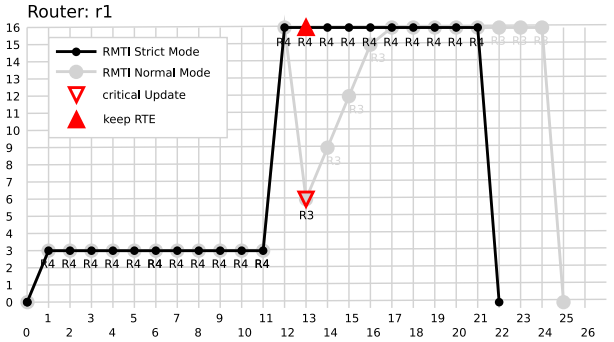

Fig. 7. Rejection of the critical update on router r1, which would have initiated a routing loop. The RMTI Strict Mode (black curve) rejects the critical update from r3 (metric 6) and keeps subnet d as unreachable, because the Y-tests fails. It computes for the Y-test: $m_A < \text{mrpm}_A + m_B \Rightarrow 6 < 3+3$ (false). The RMTI Normal Mode (gray curve in background) cannot reject this specific update.

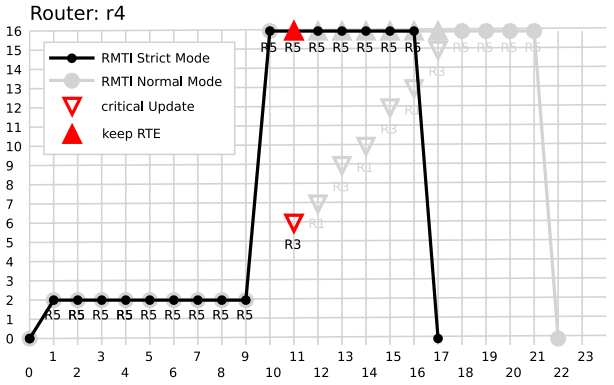

Fig. 8. The RMTI Strict Mode on router r4 also rejects the critical update from r3 (metric 6) and keeps subnet d as unreachable, because it computes for the Y-test: $m_A < \text{mrpm}_A + m_B \Rightarrow 6 < 3+2$ (false). However, the RMTI-Normal Mode on r4 also rejects all critical updates. Afterwards r1 and r4 keep the route to d with metric 16, until the garbage collection timer [RFC2453] expires and it is deleted from the routing table.

then is announced back to r4 and, moreover, also to r1. In this situation, r1 must verify the correctness of the new incoming update. The Y-test can be applied at both routers in order to recognize the Source Loop. The Strict Mode on router r1 rejects the critical update from r3 (metric 6) and keeps subnet d as unreachable, because the Y-tests fails. The test results of the Strict Mode are shown in figure 7 and 8.

The Strict Mode also performs the Y-test on Simple Loop recognition as soon as an alternative route to a network is notified by a new update. If there is an alternative route offered by a neighbor router, after the failure of the existing route, the Y-test can be used to determine if the path combination of the alternative route and the existing route is a Simple Loop. The Y-test compares the new route's metric with the sum of the mrpm entry of the new route's interface and the last valid metric of the corresponding old route. Thus, all CTI situations can be prevented. This is done by a simple rule:

If the new route passes the Y-test, it will be accepted. Otherwise it will be rejected.

Complexity: The Strict Mode only needs to access the mrpm table and, thus, we only have a linear (memory) and constant (search) complexity.

There is one special case where the Strict Mode rule rejects a valid alternative route until the timeout period for the old route to the destination has come to an end. The decision to accept the alternative route depends on the length of the smallest mrpm entry and the last valid (old) metric of the just corrupted route. It may, however, be possible that the metric of the alternative route is greater than the sum of the metric of the smallest Simple Loop and the old route (Y-test), but may be valid anyway. The update about the destination network may be an alternative route or a Source Loop. Both variants of updates show the same metric to r1 (figure 9). Thus, in this special case (and only here) the metrics are not sufficient criteria for a proper decision.

C. Careful Mode

To compensate for the decision problem mentioned above, we developed an approach to improve the convergence time of the Strict Mode, without weakening its ability to avoid CTI situations. This approach uses a timer to limit the execution time of the Strict Mode to the critical period, in which CTI situations could happen, and is called Careful Mode. It supposes that Source Loops (respective CTIs) can be deleted by the emission of a routing update within a certain time frame, whereas valid routes cannot be deleted within the same time frame. If an update of a route is rejected by the Strict Mode, this method transmits a routing update of the rejected route with metric 16 (infinity). The timer starts for a certain time frame; if a route to the same subnet with metric 16 is received on the same interface within this time frame, then the route was a Source Loop, and was just deleted on the neighbor routers. The timer is stopped and the route is kept in the routing table with metric 16. If, however, the route is continuously offered with a valid metric, then it has to be a valid route.

As soon as the timer expires, the Strict Mode is switched off for this route, and a RIP-request will be sent to the appertaining neighbor in order to achieve a consistent state

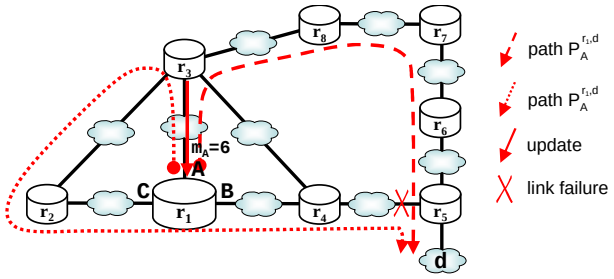

Fig. 9. This figure shows a Source Loop and an alternative path, both provided by router r3. From the view of r1, the distinct paths ($P_A^{r1,d}$) are not distinguishable, due to the same metrics. The Careful Mode can cope with this situation.

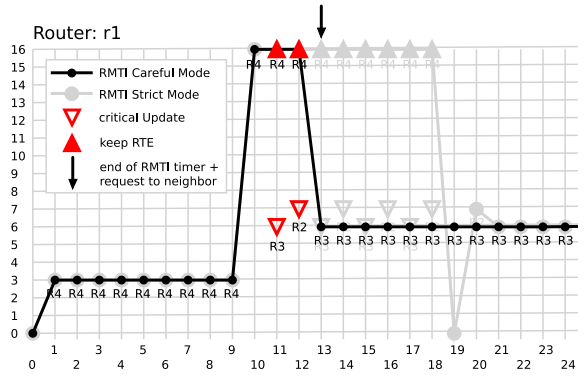

Fig. 10. This curve diagram shows the test results of the RMTI Careful Mode. It rejects the CTI update and after its timeout, shuts itself down and requests the respective neighbor concerning the doubtful route, in order to improve convergence. The RMTI Strict Mode (gray curve in background) delays a new route until the timer expired (worst case). Even if a new route is a valid replacement, the Strict Mode could deny it, because of the equal metric to a critical update, until the old route is completely deleted after the garbage collection timer expires. After that, the next corresponding update is accepted. The RMTI Careful Mode (black curve) runs only as long as the critical period of CTI emergence, and thus improves convergence significantly.

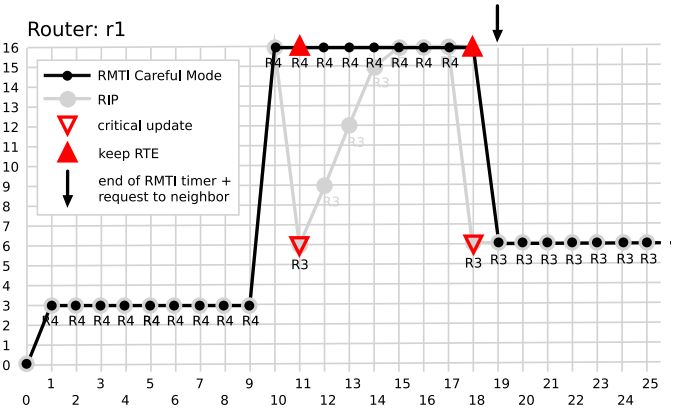

Fig. 11. Counting to infinity is avoided by the RMTI Careful-Mode (black curve). Its timer expires and the alternative route offered by router r3 is accepted. The comparison is done to standard RIP, which shows a counting to infinity situation (gray curve in background)

more fast. As the critical period for CTI situations is rather short, the timeout interval of the timer can be short too. The timeout interval could depend on the corresponding msilm value. Thus, all counting to infinity situations are handled and convergence time is improved (figure 10and 11). The Careful Mode works by a simple rule:

As long as the timer runs, it works like the Strict Mode and rejects new routes which do not pass the Y-test. After timeout, it will replace old invalid routes from the routing table with the next accordant valid route.

Complexity: As only a timer has to be implemented, no additional complexity is produced. An appropriate value for this timer can depend on the RIP update timer or on the length of the associated Simple Loop. The communication overhead remains small, because the RIP request results in only one update regarding the rejected route.

The Careful Mode based on a timer is an efficient extension for standard RIP. It copes with all counting to infinity situations and causes a minimal additional effort. The code is small, simple to handle and only the linear mrpm table is needed. A special case may occur where it cannot be sufficiently concluded from the metrics whether an update offers valid alternative routes or Source Loops. The timer-based control of the Careful Mode is the most simple and effective solution for this problem.

IV. Implementation of RMTI

Our RMTI daemon is implemented within the popular and widely used Quagga Routing Suite [12]. Thus, it runs also on routing operation systems like openwrt [18] which is a linux distribution for embedded devices.

All modes of the RMTI algorithm characterized above are implemented and tested in a network environment (figure 12).

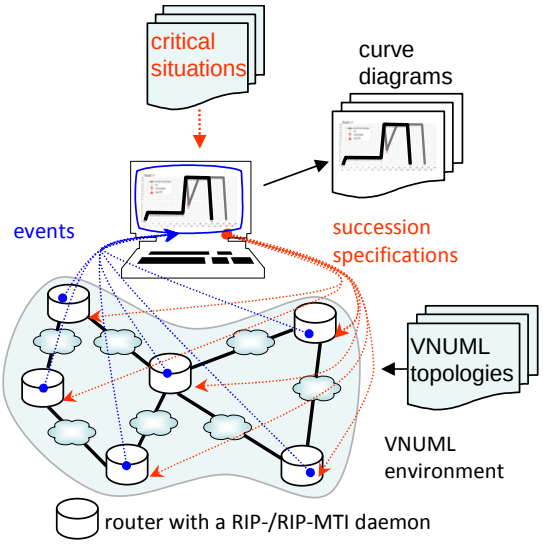

Fig. 12. The test environment is composed of a topology built up with Virtual Network User-Mode-Linux (VNUML), together with a controlling (succession specifications) and analyzing (events) extension. Each node represents one router running the Quagga Routing Suite.

Comprehensive tests are performed within virtual networks, by using the open-source virtualization tool VNUML [17], which is based on the User-Mode-Linux virtualization software [2]. This demonstrates the applicability of this approach in real network topologies and substantiates the benefits of the RMTI algorithm, especially compared to standard RIP.

In addition, we extended this daemon by a control link into our test environment. The test environment allows us to manipulate the behavior of all RIP daemons. Therefore we can trigger updates using succession specifications, permitting us to frequently test critical situations. Furthermore, the control link allows us to log all local events and information, such as interfaces and their addresses, known subnets, and every change in the local routing table. This gives us a central representation of the entire topology and available routing data for every time and location. We tested the RIP and the RMTI algorithm on a huge amount of various topologies containing loops. We verified the benefit of RMTI when it was exposed to critical situations and verified its equality in normal situations. All topologies presented in this paper are implemented using this environment and all curve diagrams presented in section IV are true output generated during the tests.

V. Related Work

Comprehensive routing loop detection was achieved by improving the distance vector protocol. This improvement was reached by designing new message formats which carried explicit topology information far beyond the simple next hop and distance information. This approach resulted in some distance vector protocols which needed more update traffic and were not compatible with the RIP-suite. The Ad hoc On-Demand Distance Vector (AODV) protocol [10] by Perkins expands the distance vector information originally based on subnet N, next hop NH and distance D, to a 4-tuple (N,NH,D,SEQ), where SEQ denotes the sequence number. Although this is provably loop-free [11], it is not compatible to the Routing Information Protocol due to the required protocol and message format changes. The Enhanced Interior Gateway Routing Protocol (EIGRP) used by Cisco is based on the DUAL algorithm [5] proposed by Garcia-Luna-Aceves. DUAL provides loop-free paths at every instance, which was proved in [5]. These protocols avoid CTI because they provide routing loop freedom, but they are not compatible with the RIPv2 standard. An approach called Source Tracing was proposed by Cheng et al [1] and Faimann [3]. Updates and routing tables provide additional information by adding a first-hop indication. Loops can be recognized recursively. The BGP [13] protocol specifies the entire path from source to destination in its routing updates in order to detect the occurrence of loops. Link-state protocols like OSPF [9] do not suffer from the CTI problem due to the use of the reliable flooding technique. Reliable flooding, however, is a technique that consumes much computational power and communication bandwidth. Every OSPF router has to maintain an up-to date version of the entire network topology. Even this enormous effort did not prevent

this protocol from suffering from transient routing loops [4], [6], [20].

Although the objectives of these techniques are basically different from our approach, it is our aim to provide a solution that has a complete and solid backward compatibility to any existing implementation of RIPv2 [8]. Our solution is a compatible extension of the RIP protocol as it uses the information provided by RIP anyway. It is even possible to deploy a new RMTI router only at selected nodes within a network.

VI. Conclusion

Routing algorithms form a real distributed system where no global state exists. Therefore, every local router has to decide locally, but each local action should contribute to global network stability.

We have demonstrated that it is possible to get new solutions to handle the routing loop problem within networks. Our approach focused on a better exploitation of update information, which is exchanged between routers anyway. Based on this approach, we found an effectual criterion to recognize and reject malicious updates as soon as they appear. Therefore, our new RMTI-algorithm improves convergence and prevents counting to infinity situations. As the forwarding tables of the new RMTI protocol do not contain wrong entries caused by critical updates, the traffic will not rotate in routing loops. This improves the stability of self-organizing router networks and grades up the scalability of networks.

The code of the RMTI implementation is small and simple to handle. RIP implementations have to be modified at a few selected points, mainly to collect information used by the RMTI algorithm. It affects the RIP implementation only within the decision-process to replace invalid routes. Most of the RMTI source code is located in an external file, which consists of a few hundred lines of code only.

Our new RMTI algorithm is possible *without* the definition of a new RIP packet format. The only modification carried out on the RIP algorithm is a more comprehensive evaluation of the common RIP update messages. This resulted in a minor computation increase within each router, which is clearly limited be the number of interfaces the router has. Our approach contains new mechanisms which form a crucial improvement to the distance vector approach.

Comprehensive testing of large networks and critical update message successions were necessary in order to gain awareness about the dynamics within network topologies and the appertaining convergence problems.

Experiments with the update timers and the maximum hop count will be done. We will offer our new protocol as a Quagga daemon for Linux in order to invite other researchers to gain experience with our new compatible routing protocol.

References

[1] C. Cheng, R. Riley, S.P.R. Kumar, and J. J. Garcia-Luna-Aceves, *A loop-free extended bellman-ford routing protocol without bouncing effect*, ACM Sigcomm Symposium Commun. Arch. and Protocols, pp. 224-236, 1989.

[2] J. Dike, *User Mode Linux*, Prentice Hall, 2006.

[3] B. Rajagopalan and M. Faiman, *A new responsive distributed shortest-path routing algorithm*, ACM Sigcomm ACM Sigcomm Symposium Commun. Arch. and Protocols, pp. 237-246, 1989.

[4] Francois, P. Bonaventure, O., *Avoiding Transient Loops During the Convergence of Link-State Routing Protocols*, Transactions on Networking, IEEE/ACM, Volume: 15, Issue: 6, Dec. 2007

[5] J.J. Garcia-Luna-Aceves, *Loop Free Routing Using Diffusing Computations*, IEEE Transactions on Networking, 1993.

[6] Hengartner, U., Moon, S., Mortier R., and Diot, C., *Detection and Analysis of Routing Loops in Packet Traces*, Proc. of 2nd Internet Measurement Workshop , Marseille, France, November 2002

[7] T. Keupen. Generation of test cases for the RIP-MTI algorithm. University of Koblenz-Landau, diploma thesis. 2007.

[8] G. Malkin, *RIP Version 2*, RFC 2453, 1998.

[9] J. Moy, *OSPF Version 2*, RFC 2328, 1998.

[10] C. E. Perkins, E. Belding-Royer, S. Das, *Ad hoc On-Demand Distance Vector (AODV) Routing*, RFC 3561, 2003.

[11] C. E. Perkins, *Ad hoc networking*, Addison-Wesley, Amsterdam 2001.

[12] Quagga home page, http://www.quagga.net, 2008

[13] Y. Rekhter, T. Li, S. Hares, *A Border Gateway Protocol 4*, RFC 4271. n

[14] A. Schmid and Ch. Steigner, *Avoiding Counting to Infinity in Distance Vector Routing*, Telecommunication Systems 19 (3-4): 497-514, March - April, 2002, Kluwer Academic Publishers.

[15] A. Schmid, O. Kandel, and Ch. Steigner, *Avoiding Counting to Infinity in Distance Vector Routing*, in Proceedings of the First International Conference on Networking (ICN 2001), Colmar, France, 2001.

[16] Ch. Steigner, H. Dickel, and T. Keupen *RIP-MTI: A New Way to Cope with Routing Loops*, in Proceedings of the Seventh International Conference on Networking (ICN 2008), Cancun, Mexico, 2008.

[17] VNUML Project home page, http://www.dit.upm.es/vnumlwiki, Technical University of Madrid (UPM), 2008.

[18] OpenWrt home page, http://www.openwrt.org, 2008

[19] Andrew S. Tanenbaum, *Computer Networks*, 3rd ed., Prentice Hall PTR, 1996

[20] Zifei Zhong, Ram Keralapura, Srihari Nelakuditi, Yinzhe Yu, Junling Wang, Chen-Nee Chuah and Sanghwan Lee, *Avoiding Transient Loops Through Interface-Specific Forwarding*, Transactions on Networking, IEEE/ACM, Volume: 15, Issue: 6, Dec. 2007

Bisher erschienen

Arbeitsberichte aus dem Fachbereich Informatik

(http://www.uni-koblenz.de/fb4/publikationen/arbeitsberichte)

Frank Bohdanovicz, Harald Dickel, Christoph Steigner, Avoidance of Routing Loops, Arbeitsberichte aus dem Fachbereich Informatik 3/2009

Stefan Ameling, Stephan Wirth, Dietrich Paulus, Methods for Polyp Detection in Colonoscopy Videos: A Review, Arbeitsberichte aus dem Fachbereich Informatik 14/2008

Tassilo Horn, Jürgen Ebert, Ein Referenzschema für die Sprachen der IEC 61131-3, Arbeitsberichte aus dem Fachbereich Informatik 13/2008

Thomas Franz, Ansgar Scherp, Steffen Staab, Does a Semantic Web Facilitate Your Daily Tasks?, Arbeitsberichte aus dem Fachbereich Informatik 12/2008

Norbert Frick, Künftige Anfordeungen an ERP-Systeme: Deutsche Anbieter im Fokus, Arbeitsberichte aus dem Fachbereicht Informatik 11/2008

Jürgen Ebert, Rüdiger Grimm, Alexander Hug, Lehramtsbezogene Bachelor- und Masterstudiengänge im Fach Informatik an der Universität Koblenz-Landau, Campus Koblenz, Arbeitsberichte aus dem Fachbereich Informatik 10/2008

Mario Schaarschmidt, Harald von Kortzfleisch, Social Networking Platforms as Creativity Fostering Systems: Research Model and Exploratory Study, Arbeitsberichte aus dem Fachbereich Informatik 9/2008

Bernhard Schueler, Sergej Sizov, Steffen Staab, Querying for Meta Knowledge, Arbeitsberichte aus dem Fachbereich Informatik 8/2008

Stefan Stein, Entwicklung einer Architektur für komplexe kontextbezogene Dienste im mobilen Umfeld, Arbeitsberichte aus dem Fachbereich Informatik 7/2008

Matthias Bohnen, Lina Brühl, Sebastian Bzdak, RoboCup 2008 Mixed Reality League Team Description, Arbeitsberichte aus dem Fachbereich Informatik 6/2008

Bernhard Beckert, Reiner Hähnle, Tests and Proofs: Papers Presented at the Second International Conference, TAP 2008, Prato, Italy, April 2008, Arbeitsberichte aus dem Fachbereich Informatik 5/2008

Klaas Dellschaft, Steffen Staab, Unterstützung und Dokumentation kollaborativer Entwurfs- und Entscheidungsprozesse, Arbeitsberichte aus dem Fachbereich Informatik 4/2008

Rüdiger Grimm: IT-Sicherheitsmodelle, Arbeitsberichte aus dem Fachbereich Informatik 3/2008

Rüdiger Grimm, Helge Hundacker, Anastasia Meletiadou: Anwendungsbeispiele für Kryptographie, Arbeitsberichte aus dem Fachbereich Informatik 2/2008

Markus Maron, Kevin Read, Michael Schulze: CAMPUS NEWS – Artificial Intelligence Methods Combined for an Intelligent Information Network, Arbeitsberichte aus dem Fachbereich Informatik 1/2008

Lutz Priese,Frank Schmitt, Patrick Sturm, Haojun Wang: BMBF-Verbundprojekt 3D-RETISEG Abschlussbericht des Labors Bilderkennen der Universität Koblenz-Landau, Arbeitsberichte aus dem Fachbereich Informatik 26/2007

Stephan Philippi, Alexander Pinl: Proceedings 14. Workshop 20.-21. September 2007 Algorithmen und Werkzeuge für Petrinetze, Arbeitsberichte aus dem Fachbereich Informatik 25/2007

Ulrich Furbach, Markus Maron, Kevin Read: CAMPUS NEWS – an Intelligent Bluetooth-based Mobile Information Network, Arbeitsberichte aus dem Fachbereich Informatik 24/2007

Ulrich Furbach, Markus Maron, Kevin Read: CAMPUS NEWS - an Information Network for Pervasive Universities, Arbeitsberichte aus dem Fachbereich Informatik 23/2007

Lutz Priese: Finite Automata on Unranked and Unordered DAGs Extented Version, Arbeitsberichte aus dem Fachbereich Informatik 22/2007

Mario Schaarschmidt, Harald F.O. von Kortzfleisch: Modularität als alternative Technologie- und Innovationsstrategie, Arbeitsberichte aus dem Fachbereich Informatik 21/2007

Kurt Lautenbach, Alexander Pinl: Probability Propagation Nets, Arbeitsberichte aus dem Fachbereich Informatik 20/2007

Rüdiger Grimm, Farid Mehr, Anastasia Meletiadou, Daniel Pähler, Ilka Uerz: SOA-Security, Arbeitsberichte aus dem Fachbereich Informatik 19/2007

Christoph Wernhard: Tableaux Between Proving, Projection and Compilation, Arbeitsberichte aus dem Fachbereich Informatik 18/2007

Ulrich Furbach, Claudia Obermaier: Knowledge Compilation for Description Logics, Arbeitsberichte aus dem Fachbereich Informatik 17/2007

Fernando Silva Parreiras, Steffen Staab, Andreas Winter: TwoUse: Integrating UML Models and OWL Ontologies, Arbeitsberichte aus dem Fachbereich Informatik 16/2007

Rüdiger Grimm, Anastasia Meletiadou: Rollenbasierte Zugriffskontrolle (RBAC) im Gesundheitswesen, Arbeitsberichte aud dem Fachbereich Informatik 15/2007

Ulrich Furbach, Jan Murray, Falk Schmidsberger, Frieder Stolzenburg: Hybrid Multiagent Systems with Timed Synchronization-Specification and Model Checking, Arbeitsberichte aus dem Fachbereich Informatik 14/2007

Björn Pelzer, Christoph Wernhard: System Description:“E-KRHyper“, Arbeitsberichte aus dem Fachbereich Informatik, 13/2007

Ulrich Furbach, Peter Baumgartner, Björn Pelzer: Hyper Tableaux with Equality, Arbeitsberichte aus dem Fachbereich Informatik, 12/2007

Ulrich Furbach, Markus Maron, Kevin Read: Location based Informationsystems, Arbeitsberichte aus dem Fachbereich Informatik, 11/2007

Philipp Schaer, Marco Thum: State-of-the-Art: Interaktion in erweiterten Realitäten, Arbeitsberichte aus dem Fachbereich Informatik, 10/2007

Ulrich Furbach, Claudia Obermaier: Applications of Automated Reasoning, Arbeitsberichte aus dem Fachbereich Informatik, 9/2007

Jürgen Ebert, Kerstin Falkowski: A First Proposal for an Overall Structure of an Enhanced Reality Framework, Arbeitsberichte aus dem Fachbereich Informatik, 8/2007

Lutz Priese, Frank Schmitt, Paul Lemke: Automatische See-Through Kalibrierung, Arbeitsberichte aus dem Fachbereich Informatik, 7/2007

Rüdiger Grimm, Robert Krimmer, Nils Meißner, Kai Reinhard, Melanie Volkamer, Marcel Weinand, Jörg Helbach: Security Requirements for Non-political Internet Voting, Arbeitsberichte aus dem Fachbereich Informatik, 6/2007

Daniel Bildhauer, Volker Riediger, Hannes Schwarz, Sascha Strauß, „grUML – Eine UML-basierte Modellierungssprache für T-Graphen“, Arbeitsberichte aus dem Fachbereich Informatik, 5/2007

Richard Arndt, Steffen Staab, Raphaël Troncy, Lynda Hardman: Adding Formal Semantics to MPEG-7: Designing a Well Founded Multimedia Ontology for the Web, Arbeitsberichte aus dem Fachbereich Informatik, 4/2007

Simon Schenk, Steffen Staab: Networked RDF Graphs, Arbeitsberichte aus dem Fachbereich Informatik, 3/2007

Rüdiger Grimm, Helge Hundacker, Anastasia Meletiadou: Anwendungsbeispiele für Kryptographie, Arbeitsberichte aus dem Fachbereich Informatik, 2/2007

Anastasia Meletiadou, J. Felix Hampe: Begriffsbestimmung und erwartete Trends im IT-Risk-Management, Arbeitsberichte aus dem Fachbereich Informatik, 1/2007

„Gelbe Reihe“
(http://www.uni-koblenz.de/fb4/publikationen/gelbereihe)

Lutz Priese: Some Examples of Semi-rational and Non-semi-rational DAG Languages. Extended Version, Fachberichte Informatik 3-2006

Kurt Lautenbach, Stephan Philippi, and Alexander Pinl: Bayesian Networks and Petri Nets, Fachberichte Informatik 2-2006

Rainer Gimnich and Andreas Winter: Workshop Software-Reengineering und Services, Fachberichte Informatik 1-2006

Kurt Lautenbach and Alexander Pinl: Probability Propagation in Petri Nets, Fachberichte Informatik 16-2005

Rainer Gimnich, Uwe Kaiser, and Andreas Winter: 2. Workshop "Reengineering Prozesse" – Software Migration, Fachberichte Informatik 15-2005

Jan Murray, Frieder Stolzenburg, and Toshiaki Arai: Hybrid State Machines with Timed Synchronization for Multi-Robot System Specification, Fachberichte Informatik 14-2005

Reinhold Letz: FTP 2005 – Fifth International Workshop on First-Order Theorem Proving, Fachberichte Informatik 13-2005

Bernhard Beckert: TABLEAUX 2005 – Position Papers and Tutorial Descriptions, Fachberichte Informatik 12-2005

Dietrich Paulus and Detlev Droege: Mixed-reality as a challenge to image understanding and artificial intelligence, Fachberichte Informatik 11-2005

Jürgen Sauer: 19. Workshop Planen, Scheduling und Konfigurieren / Entwerfen, Fachberichte Informatik 10-2005

Pascal Hitzler, Carsten Lutz, and Gerd Stumme: Foundational Aspects of Ontologies, Fachberichte Informatik 9-2005

Joachim Baumeister and Dietmar Seipel: Knowledge Engineering and Software Engineering, Fachberichte Informatik 8-2005

Benno Stein and Sven Meier zu Eißen: Proceedings of the Second International Workshop on Text-Based Information Retrieval, Fachberichte Informatik 7-2005

Andreas Winter and Jürgen Ebert: Metamodel-driven Service Interoperability, Fachberichte Informatik 6-2005

Joschka Boedecker, Norbert Michael Mayer, Masaki Ogino, Rodrigo da Silva Guerra, Masaaki Kikuchi, and Minoru Asada: Getting closer: How Simulation and Humanoid League can benefit from each other, Fachberichte Informatik 5-2005

Torsten Gipp and Jürgen Ebert: Web Engineering does profit from a Functional Approach, Fachberichte Informatik 4-2005

Oliver Obst, Anita Maas, and Joschka Boedecker: HTN Planning for Flexible Coordination Of Multiagent Team Behavior, Fachberichte Informatik 3-2005

Andreas von Hessling, Thomas Kleemann, and Alex Sinner: Semantic User Profiles and their Applications in a Mobile Environment, Fachberichte Informatik 2-2005

Heni Ben Amor and Achim Rettinger: Intelligent Exploration for Genetic Algorithms – Using Self-Organizing Maps in Evolutionary Computation, Fachberichte Informatik 1-2005