Improving Convergence Time of Routing Protocols

Götz Lichtwald, Uwe Walter
and Martina Zitterbart
Institute of Telematics
University of Karlsruhe, Germany
Email: {lichtwald, walter, zit}@tm.uka.de

***Abstract*— One of the main design goals of the Internet is robustness against failures. Normally, this is accomplished by redundancy and dynamic routing protocols that automatically adapt to failures: If a link is unavailable, data packets can generally be sent via alternative paths. An essential requirement for this is a fast mechanism for failure detection, since routing protocols can only start to reroute traffic around problems as soon as they get aware of them. This paper proposes a novel design of a generic failure detection service to be utilized by routing protocols that aims at dramatically decreasing the detection times of today's mechanisms. After the introduction of the concept and its evaluation, the integration of the new failure detection service into BGP is described. Furthermore, an example for the adaption of a routing protocol using the proposed service is given.**

I. Introduction

Data packets traveling through the Internet typically traverse multiple routers and, thus, multiple physical links interconnecting them. Whenever such a link fails, dynamic routing protocols try to provide an alternative path towards the destination. For this task, it is crucial that the routing protocol quickly detects such a link failure. Especially with the increasing use of the Internet for mission critical applications any unnecessary loss of connectivity can hardly be tolerated and has to be kept as short as possible.

One of the fastest possibility for link failure detection can be achieved by co-operation with the lower (link level) layers. If they are able to detect a link breakdown in hardware, e.g. by the loss of the physical or optical signal, they can immediately notify the network layer about the failure. This is actually already deployed, but has some shortcomings. Fixing implementation errors in router operating systems [1] that inhibit a quick notification may help sometimes. Link level failure detection in general is not always possible.

For example, in an environment where switches are involved in the router interconnection, the possibility that links may fail behind such a switch, prevents the chance for a fast link level failure detection.

This is why existing routing protocols, like Routing Information Protocol (RIP), Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS) or Border Gateway Protocol (BGP) typically exchange periodic messages to check whether their peer is still reachable and alive, as described in section II. These periodic test messages, e.g. *KEEPALIVE* or *HELLO* messages, however, consume bandwidth and processing time. Because of these reasons, sometimes there are no dedicated check packets, but normal routing protocol traffic takes this function and is used as regular alive beacon.

In order to further limit resource consumption, rather long time intervals between consecutive check messages have been chosen. This also helps to reduce the number of so-called false positives. Whenever a link is wrongly declared to be broken, due to several short-time link noise periods or temporarily router processor overload this is called a false positive.

It is a permanent tradeoff that must be found between a delayed failure detection and too many false positives. If a link breakdown has occurred, but has not yet been recognized, packets sent via the defective link get inevitably lost during that period. On the other side, during every false positive, whenever an operational link is mistakenly declared to be down, packets are unnecessarily rerouted and other routers are notified of the alleged failure, leading to routing fluctuations and instabilities.

Considering the very high speed links, the powerful routers as well as the low bit error rates within today's Internet, long time intervals between periodic check messages do not appear to be appropriate any longer. Liveness check mechanisms of routing protocols—especially the timeout value—need to be updated appropriately.

This is in accordance recent efforts of with major routing vendors (i.e. Juniper) as well as with current standardization efforts within the IETF (cf. section V). Common to all these efforts is the goal to improve, i.e. to shorten, the link failure detection time.

This paper goes a step further and does not only propose to change the default values between periodic check messages, but develops a novel generic service for failure detection, called Adjacent Peer Check Service (APCS) [2] that enables any routing protocol to detect link outages faster than before (see section III. Furthermore, does the APCS not only check the physical reachability but also the operational state of the control plane and can be integrated into existing routing protocols. So, in contrast to the efforts of the IETF [3] and major routing vendors, the Adjacent Peer Check Service is designed to improve existing routing protocols.

As the APCS is a generic and routing protocol independent service, network operators can define the peer check time intervals for the check messages depending on their network demands. This means that the check message interval for WLAN connections can be set to a different value than for LAN connections to accommodate

to the different bandwidths and loss rates. Furthermore, it is possible to define the threshold of check message losses until a link is declared to be broken.

As mentioned before, those time intervals can be easily adjusted to the physical network environment conditions, e.g. fiber, radio or coax on the one hand side. On the other hand, those time intervals can be adapted to the changing demands of a connection, i.e. a connection carries more high priority traffic and in case a failure occurs the outage has to be kept as short as possible.

This paper will provide the Adjacent Peer Check Service protocol design including its evaluation in several test-bed scenarios (see section IV). Those scenarios compare the advantages that can be achieved by extending currently deployed routing protocols like RIP, OSPF and BGP, with improvements achieved by the Adjacent Peer Check Service. The evaluation also took highly loaded links into consideration to prove that the novel approach also works in congested networks.

Furthermore, an analysis of how the currently deployed routing protocols can be improved by the Adjacent Peer Check Service is provided. The paper concludes in section VI with a detailed description how BGP can be improved with the novel Adjacent Peer Check Service. Further a detailed analysis about the improvements that would come along with this BGP extension, concerning the inter-domain convergence time, is given.

II. Failure Detection Mechanisms in Routing Protocols

In the following, an overview is given about the mechanisms that are used in some widely-deployed routing protocols to detect failures, for example link or router outages, which trigger the process of finding alternative paths. This will demonstrate the similarities between all these protocols and help in understanding the concept of the Adjacent Peer Check Service, described in section III.

A. Routing Information Protocol

The *Routing Information Protocol (RIP)*, developed in 1988 it [4] is an example of a *Distance Vector Protocol*. A distance vector consists of a destination address and a metric that represents the cost for reaching this destination. RIP exchanges such distance vectors between all routers to allow them to calculate the optimal paths to all possible destination nodes.

All necessary RIP messages are transmitted as UDP datagrams on port 520. If there are no reasons to exchange new distance vectors or answers to incoming requests, regular updates are sent every 30 seconds. Only adjacent routers exchange RIP messages directly with each other, which allows them to detect failures in the communication with their neighbor. If there has been no incoming message of an adjacent router for a time period of 180 seconds, the corresponding router is assumed to be unreachable, an event which declares all routes leading via this link invalid. This point in time marks the beginning of the rerouting process, as the remaining neighbor routers are notified of the broken connection by sending out new distance vectors.

B. Open Shortest Path First

The *Open Shortest Path First (OSPF)* protocol [5] was created by the OSPF working group of the IETF as an IP-based routing protocol to be used inside Autonomous Systems (AS). OSPF uses a Link State Database that describes the topology of the AS, inside of which it is deployed. To synchronize this database consistently among all OSPF-speaking routers, the contained information is flooded throughout the whole AS via so-called *Link State Announcements (LSA)*. This allows all routers to build the same Link State Database and to calculate the shortest paths to all possible destinations on their own.

Different sorts of LSAs exist that are all transmitted via IP packets carrying the protocol number 89. The most important OSPF packet type is the *HELLO* packet, used for automatic detection of neighbors and failure detection. *HELLO* packets are broadcasted by every router via each of its interfaces in regular time intervals called *Hello Interval*. Communication to a router is declared to be broken, if there has been no *HELLO* packet received from it for another important time interval, named *Router Dead Interval*. Both time intervals are included in every *HELLO* packet and are, therefore, identical for all OSPF routers inside the AS. The default values, proposed in [5], for the *Hello Interval* are 10 seconds for local area networks (LANs) and 30 seconds for wide area networks (WANs). The same document recommends to set the *Router Dead Interval* to four times the *Hello Interval*. This means that a failure is detected after a maximum time of 40–120 seconds, before the process of finding alternative paths can be started.

C. Border Gateway Protocol

The *Border Gateway Protocol (BGP)* [6] is the standard inter-AS routing protocol deployed between all Autonomous Systems of the Internet. It is used to exchange information between all BGP-speaking routers about the reachability of destination networks in form of *Path Vectors*, i.e., BGP is a so-called path vector routing protocol. Essentially, these path vectors consist of a specific AS-path towards a destination network (represented by an IP prefix) and are exchanged via TCP connections on port 179 between adjacent routers. Using all received path vectors, each BGP router can build its own routing information base and calculate the best paths towards all reachable destination networks from its point of view.

If no path vector messages are exchanged, a BGP connection between adjacent routers is held open by so-called *KEEPALIVE* packets that are sent periodically. The connection is torn down, if there has been no incoming notification for a time interval, called *Hold Time*. The specification [6] recommends a maximum spacing of one third of the Hold Time between two *KEEPALIVE* packets. The Hold Time is negotiated between two adjacent routers during their BGP connection setup and must be no shorter

Protocol	Missing Data	Default Timer Setup
RIP	Distance Vector Packet	180 seconds
OSPF	*HELLO* Packet	40–120 seconds
BGP	*KEEPALIVE* Packet	90 seconds

TABLE I
COMPARISON OF FAILURE DETECTION MECHANISMS IN WIDELY-DEPLOYED ROUTING PROTOCOLS.

than three seconds[1]. A default value of 90 seconds is proposed in [6]. In the worst-case, this is the time, BGP needs to detect a communication failure and to send out corresponding notifications to all neighbor routers, allowing them to direct their traffic over alternative paths.

D. Comparison

All the presented routing protocols use a similar mechanism to detect communication failures by waiting for missing incoming messages for a certain amount of time.

The similarities of the deployed failure detection mechanisms (cf. table I) open up the opportunity to extract the peer checking mechanism and define a general architecture. This architecture can be used to design a failure detection service that can be utilized by appropriately adapted routing protocols. Such an approach is followed in section III.

III. ADJACENT PEER CHECK APPROACH

Section II provides an overview of two inherent properties of recent routing protocols. The routing protocols RIP, OSPF and BGP have a mechanism to detect failures of their peers. A further inherent property is the frequency that is used to check peers—depending on the network conditions. The mechanism, the routing protocols use for checking, depends on the environment they were designed for. The above mentioned protocols RIP and OSPF cover the intra-domain area and BGP is a representative of the inter-domain area.

Looking at the mentioned routing protocols from a historic point of view, RIP was one of the very first routing protocols. The driving factor at this time was to provide connectivity with the limiting condition of bandwidth scarcity and the—at least sometimes—poor quality of physical lines that lead to link outages. Nowadays the driving factor is not only connectivity but also resilience. An example are the efforts that are taken by a big router vendor to push OSPF convergence to the order of magnitude of less than a second. Also the standardization body IETF, with its draft about Bidirectional Forwarding Detection [7], provides a contribution (see section V).

A. Basic Concept

The Adjacent Peer Check approach alleviates the fact that routing protocols are designed only for a certain network environment and are less adaptive to changing circumstances.

[1]However, the transmission of *KEEPALIVE* packets can be switched off completely by setting the Hold Time to zero.

The Adjacent Peer Check Service provides a peer check service shim layer below the routing layer, so that the peer check mechanism does not have to be an inherent part of the routing protocol.

The design of the protocol is kept simple and robust. The Adjacent Peer Check Service periodically issues check messages towards its peers.

The diagram in fig. 1 depicts, from the point of view of *Router A*, how the check messages are sent. The so-called *Check Interval* is a parameter that is setup depending on the network circumstances. This parameter defines the time spacing between two sent check messages. What is not shown in fig. 1, is the point of view of *Router B*. *Router A* receives check messages from *Router B* as well. Those check messages from *Router A* and *Router B* do not have to be correlated. *Router A* detects a check message from *Router B* and starts a timer that assures that—within a certain period of time—the next check messages is received or that a failure is indicated towards the routing protocol instance. This indication is done by an event. A possible event could be a *Peer-Down-Event*.

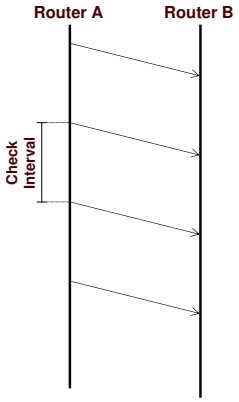

Fig. 1. Asynchronous Peer Check

It would also have been possible to design the protocol in a stop and wait manner. Which means that each check message would have to be confirmed. In order to reduce this kind of interaction and to keep the protocol simple and robust the above presented design was chosen.

The design of the Adjacent Peer Check Protocol requires another property: The check messages must be routable. A scenario for routable check messages is I-BGP. For I-BGP it is not required that routers are directly connected, which may bring up the case that I-BGP messages have to be routed by, e.g. an OSPF router to reach their destination. Routing is needed, if the Adjacent Peer Check Service wants to check the liveness of the control plane of an I-BGP peer. In this case the check message must be routable. This was the reason why the payload of IPv4 [8] packets is used to transport the check message towards its destination.

B. Architecture

A framework was developed so that the Adjacent Peer Check Service could be used by any routing protocol. This framework is depicted in fig. 2.

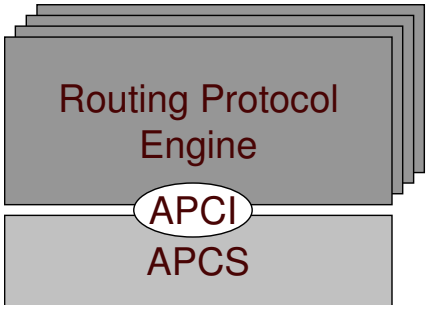

Fig. 2. Routing Architecture Framework

The framework provides an interface between the well-known routing protocol instances and the Adjacent Peer Check Service. The lower part of the framework (cf. fig. 2) accomplishes the task of checking the liveness of peers and its routing protocol instances. As mentioned, the communication between the two layers is provided by an interface, the so-called Adjacent Peer Check Interface (APCI), which is depicted in fig. 2. This interface provides several methods that the routing protocol instance can use to gain information about its sessions with its peers. Furthermore, the routing protocol instance is triggered via this Adjacent Peer Check Interface, in case a peer or its routing protocol instance seems to be down or not reachable.

C. Adaption of RIP, OSPF and BGP

To use the Adjacent Peer Check Service the routing protocol has to register itself at the service. After the routing protocol instance has registered itself at the APCS, the routing protocol instance is informed about its peers' status via the Adjacent Peer Check Interface. Several events are defined for this task:

- *Peer-Down-Event*: Indicates that a certain peer is down.
- *Peer-Up-Event*: Indicates that a certain peer is up again.
- *Protocol-Down-Event*: Indicates that the peer is up, but the peer's protocol instance is out of service.
- *Protocol-Up-Event*: Indicates that the peer is still up and that the peer's protocol instance has recovered.
- *Graceful-Restart-Event*: Indicates that the peer's protocol instance is down for maintenance, but the forwarding of packets is still in service.

1) RIP: In case a *Peer-Down-Event* or *Protocol-Down-Event* occurs, the weights for routes, leading over this peer, get a weight of 16. This special weight of 16 indicates that the route is not in service any longer [4]. Additionally, the so-called *Garbage Collection Timer* has to be started.

In case the RIP routing protocol instance receives the *Peer-Up-Event* or *Peer-Protocol-Event* no special action has to be taken. The reason therefore is that RIP deletes invalid routes, so in case the Adjacent Peer Check Service indicates a route as valid again, the RIP routing protocol instance can not fall back on the deactivated route, but RIP has to recalculate this route. In order to prevent false down declaration of a peer the *timeout* timer should get higher values during the *Graceful Restart* process.

2) OSPF: In case OSPF receives a *Peer-Down-Event* or a *Protocol-Down-Event* from the Adjacent Peer Check Service this is equivalent to a missing *HELLO* packet after a so-called *Router Dead Interval*. The Adjacent Peer Check Service can not substitute the periodic *HELLO* messages, but can be used as improvement for a faster failure detection.

In case the Adjacent Peer Check Service indicates a *Peer-Up-Event* or *Protocol-Up-Event* the OSPF routing engine handles this as if a new router starts sending *HELLO* messages.

If the Adjacent Peer Check Service indicates a *Graceful-Restart-Event* the *Router Dead Interval* should be temporarily prolonged to prevent a false down detection of the peer.

3) BGP: The Border Gateway Protocol uses periodically sent *KEEPALIVE* messages to check the liveness of a peer. The so-called *Hold Timer* defines the frequency of this check. This setup may be used for the Adjacent Peer Check Service instead of using the *KEEPALIVE* messages.

In case a *Peer-Down-Event* or a *Protocol-Down-Event* is received, BGP tears down the session. The occurrence of a *Peer-Up-Event* or a *Protocol-Up-Event* causes the setup procedure of a BGP session.

In case a *Graceful-Restart-Event* is received the session is kept open for a certain period of time [9]. It is assumed that the peer still forwards packets and that only the control plane is rebooting.

D. Improvements

As it was shown, only small changes have to be made to use the Adjacent Peer Check Service. So, existing routing protocols can use the new approach to improve their failure detection time. A further advantage is that routing protocols—using the Adjacent Peer Check Service—are adaptive in terms of physical connection conditions. Furthermore, a failure on a mission critical link can be detected much more faster than common routing protocols do currently.

The Adjacent Peer Check Service enables BGP, for example, to react more adequate on short time physical line outages of so-called Stub ASes (see also section VI).

IV. Evaluation

After the basic concept of the Adjacent Peer Check approach was described in section III, it is proved that this approach works. First, it is explained what and how the measurements are done and furthermore a performance analysis is given that shows the efficiency of the Adjacent Peer Check approach.

A. Validation

In order to validate the Adjacent Peer Check approach two basic scenarios where examined. The first scenario (see section IV-A.1) consists of two routers that "talked" directly to each other. This represents the common BGP

setup, where two BGP peers are directly connected via a physical link. As this approach causes no interference with other peers, it is enough to look just at two peers.

The second scenario (see section IV-A.2) represents a network structure that requires a routing for I-BGP messages (see III-A). For this scenario a three router setup was chosen. Two of those routers check each other using the Adjacent Peer Check Service and the third router has only routing functionality and is not Adjacent Peer Check Service aware.

To prove the correctness of the test results, the range, i.e. minimum and maximum, of the failure detection delay was theoretically calculated. The minimum and maximum delay concerning the detection of a link outage or a control plane failure can be described as:

$$\begin{aligned} \mathrm{DetectionDelay}_{\min} &= \mathrm{PeerDownIntvl} - \mathrm{CheckIntvl} \\ \mathrm{DetectionDelay}_{\max} &= \mathrm{PeerDownIntvl} \end{aligned}$$

The theoretical minimum and maximum failure detection time can be calculated as follows (cf. fig. 3): The earlier the last check message is sent from router 2, the earlier router 1 is able to detect the failure. The earliest point of time when a check message could have been sent is the time the router crashes—called time X—minus the *Check Interval*. Router 1 can first detect the failure after the time of the last check message plus the *Peer-Down-Interval*. Neglecting the transmission delay, the point of time D, where the failure is detected can be calculated as follows:

$$\begin{aligned} D &= X - \mathrm{CheckIntvl} + \mathrm{PeerDownIntvl} \\ &= X + (\mathrm{PeerDownIntvl} - \mathrm{CheckIntvl}) \end{aligned}$$

The latest possible point of time for D is correlated to the time when the last check message was sent. In this case the check message was sent at the time the failure occured. Again neglecting the transmission delay, the point of time, where the failure is detected can be calculated as follows:

$$D = X + \mathrm{PeerDownIntvl}$$

1) First Scenario: The first test scenario consists of only two routers that where directly connected. One of those routers was configured to randomly tear down the connection, which is equivalent to a link outage. The results of a 100 Mbit/s link are partly depicted in fig. 4. The *Check Interval* was varied between [5 ms, 10 ms, ..., 100 ms]. For each of those *Check Intervals* ten runs were performed. The graph in fig. 4 depicts the maximum, minimum and the average value of those ten runs. During those runs the *Peer-Down-Interval* was three times as big as the *Check Interval*.

The results of this setup have shown that the Adjacent Peer Check Service works as specified. For very small *Check Intervals* it could be observed that some measured values are below the theoretical minimum. This is due to the inaccuracy of the timer of the used Linux operating system. It turned out that currently, Linux is not able to handle shorter time intervals than 10 ms. This fact

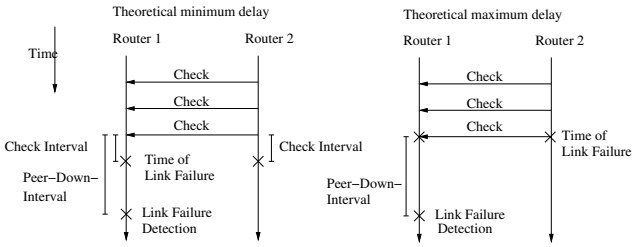

Fig. 3. Theoretical minimum and maximum failure detection delay

generated an operating system inherent limitation to our test. So tests that ran with a 5 ms setup had a de-facto 10 ms behavior. The only difference between the 5 ms and 10 ms setup that could be observed was the *Peer-Down-Interval*. For the 5 ms setup it was 15 ms and for the 10 ms setup it was 30 ms.

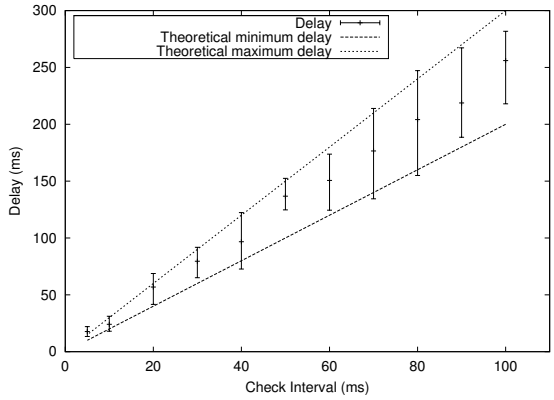

Fig. 4. Theoretical and measured detection delays with *Check Intervals* between 5 ms and 100 ms

2) Second Scenario: Figure 5 shows the test results of the same test-bed setup with the difference that the Adjacent Peer Check messages are routed. This router also is an off-the-shelf Linux box that had no other task than to route traffic. The Linux Kernel that was used was 2.4.x and the link speed was 100 Mbit/s.

The results of this test are depicted in fig. 5 for the *Check Intervals* [5 ms, ..., 100 ms]. Those results are—as expected—almost the same as the results from the non-routing-setup. This proves that the Adjacent Peer Check Service also works in scenarios, where check messages have to be routed, e.g. for I-BGP (cf. section III-A).

B. Performance

After the basic validation of the functionality, the more important question is, how well the Adjacent Peer Check Service will perform under more realistic conditions, for example a heavily loaded or even overloaded network.

Therefore, we investigated the influence of network traffic on the efficiency of the APCS. Theoretically, the occurrence of big traffic bursts has the worst impact on periodic check packets: For example one packet could be sent immediately because there is no other traffic queued, while the next check packet could encounter a rather full

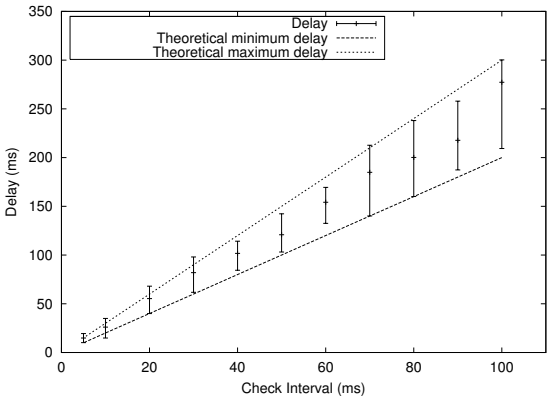

Fig. 5. Theoretical and measured detection delays with *Check Intervals* between 5 ms and 100 ms in routed scenario

waiting queue due to an incoming traffic burst and get significantly delayed. This could lead to false positives, if the configuration is chosen too tight.

To avoid such a behavior, appropriate *Peer-Down-Intervals* must be identified for the desired setup. This was demonstrated in the same test setup as described in section IV-A, but this time with heavy traffic bursts (maximum UDP traffic possible) on the connection.

The valid minimum *Peer-Down-Intervals* depending on the *Check Intervals* are depicted in figure 6 in a slightly different form, namely as factor of the *Check Interval*: *Peer-Down-Interval = Peer-Down-Factor · Check Interval*.

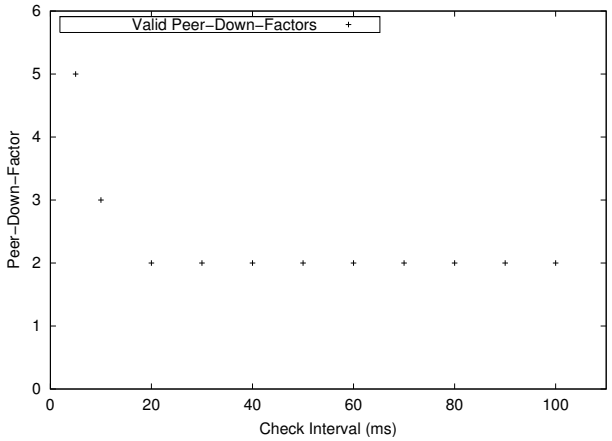

Fig. 6. Valid Peer-Down-Factors under maximum network load

It can be seen, that for most *Check Intervals*, the smallest possible *Peer-Down-Factor* equals 2. This means that at most a single check packet can get lost (or heavily delayed) before a failure condition would be detected, which would be a false positive in this scenario, since the link was not interrupted, but only highly loaded.

For the lowest possible *Check Intervals*, higher *Peer-Down-Factors* are necessary to avoid false positives. This is due to tighter timing conditions. If the *Check Interval* is only 10 ms, with a *Peer-Down-Factor* of 2, a failure is detected after there has been no incoming check packet for only 20 ms. Besides, check packets being possibly delayed that long in waiting queues with volatile lengths, again this setup hit technical limits of the Linux operating system used for the demonstration.

The network load did not have too much influence on the APCS, which can be explained by looking at the load caused by the Adjacent Peer Check Service itself (as an example, it is assumed that a standard Ethernet is used, simplified by leaving special timings aside, like e.g. interframe spacings). The check packets are small enough to fit into a minimum Ethernet frame of 64 bytes. This fact alone gives the check packets a desirable advantage over longer data packets when it comes to insertion into an overloaded waiting queue, as it is more probable that there is enough room for a very small check packet than any bigger one. The bandwidth consumption of an unidirectional Peer-Check association $BW_{\mathrm{Peer-Check}}$ can be calculated by multiplying the minimum frame size with the frequency of the check packets $\mathrm{freq}_{\mathrm{CheckPackets}}$:

$$\begin{aligned} BW_{\mathrm{Peer-Check}} &= 64\,\mathrm{bytes} \cdot \mathrm{freq}_{\mathrm{CheckPackets}} \\ &= 64\,\mathrm{bytes} \cdot \frac{1}{\mathrm{CheckInterval}} \end{aligned}$$

Even for a *Check Interval* of only 10 ms, this means, that only a bandwidth of 51,2 kilobit per second of the available capacity is used by the APCS, leaving much room for smaller *Check Intervals* if there are no implementation limits.

C. Configuration

As it has been shown in the previous section, the configuration of the Adjacent Peer Check Service is a critical issue, like with all failure detection mechanisms that use periodic check packets. Tweaking towards a faster failure reaction can easily lead to increasing false positive rates.

To demonstrate this problem, additional tests with the APCS have been done in a WLAN network with a packet loss rate of more than 1 %, while in the setup used in section IV-A.1 there has hardly been *any* packet loss.

Such a scenario can take a significant amount of time until a configuration is found that does not have too many of the disadvantages already mentioned. For example, with 20 ms *Check Interval*, even a *Peer-Down-Factor* of 6 led to several false positives. Using a *Check Interval* of 100 ms also produced some (wrong) Peer-Down Events with a *Peer-Down-Factor* of 2. Increasing the latter to 3 improved the situation under good signal conditions. When the wireless reception degraded, this was not enough anymore and false positives were generated again. Finally, using a *Peer-Down-Factor* of 4 (still with the *Check Interval* 100 ms) solved the problems in our test setup and worked successfully. However, the situation might change at any time and make new adaptions necessary.

Apparently, finding just the right configuration values that avoid false positives and still detect real failures quickly can become quite difficult. In future work, the

developed APCS framework will be equipped with mechanisms to help achieve this task. Through the modular framework, routing protocols equipped with an interface to the Adjacent Peer Check Service, will be able to take advantage of these possible improvements, relieving their operators from endless configuration tests.

V. Related Work

The fact that most default settings of failure detection mechanisms in currently deployed routing protocols are inappropriate for nowadays networks has led to more research in this area.

The convergence time of the IS-IS routing protocol [10] has been studied in [11]. Similar to the mechanisms, presented in section II, IS-IS also uses a hello protocol to detect failures. If a fixed number of *HELLO* packets from an adjacent router are missed, the adjacency is declared down. The default values, observed by the authors in [11], are ten second intervals between consecutive *HELLO* packets and a number of three lost packets before IS-IS assumes a failure condition. Those times sums up to a total maximum of 30 seconds until a failure is reliably detected (cf. table I). In the following, it is proposed to change the granularity of the hello interval to milliseconds, since there is no technical reason against such an improvement.

The Internet Engineering Task Force has also picked up the topic and begun work an a protocol to determine the liveness of routing protocols, called *Protocol Liveness Protocol (PLP)* [3]. It also aimed at having a simple mechanism for liveness checks to be used by all protocols and allow fast detection of failures.

Recently, a new design became the successor of PLP, sharing the same goals. The new protocol is called *Bidirectional Forwarding Detection (BFD)* [7] and is intended to quickly detect faults in the bidirectional path between two forwarding engines, independent from the deployed data and routing protocols.

The development of the Adjacent Peer Check was started before the first version of the PLP draft has been published. Looking at PLP, there were some design problems like the vector indicating the status of the routing protocol instance, e.g., RIP, OSPF or BGP. Because this status information was carried in every data packet, it caused an unnecessary overhead. Furthermore, the Type-Length-Value (TLV) structure is processing expensive [12]. So, the working group decided to do a redesign and came up with the BFD draft. This draft has a lot in common with our Adjacent Peer Check approach. The size of the check messages is kept small, so that no unnecessary amount of data is transmitted. But in contrast to BFD, the Adjacent Peer Check approach aims at a more academic goal. The APC can be seen as first part of a new routing architecture where the Adjacent Peer Check represents one module of this architecture. A further module could be the routing algorithm and another module for either a path or distance vector routing approach.

VI. Combining BGP and APCS

The Adjacent Peer Check Service can provide a significant improvement to the convergence time of BGP. With the Adjacent Peer Check Service BGP can detect link failures or the malfunction of its peer's control plane much faster.

A. How to extend BGP

In general trying to modify an existing and established protocol is difficult, as most of the installations can only be changed by a firmware- or software-upgrade. So the Adjacent Peer Check approach (cf. fig. 2) tries to offer an integration scheme that does not need a so-called "flag day". The Adjacent Peer Check capability can slowly be integrated and provides benefits, if it is used. But on the other hand, it does not harm anything if the Adjacent Peer Check capability is disabled or not installed.

To accomplish the integration of the Adjacent Peer Check Service, the *Hold Timer* of the Border Gateway Protocol is set to zero. This implies that no *KEEPALIVE* messages are exchanged between the BGP peers [6]. This setup is used to substitute the *KEEPALIVE* messages with the Adjacent Peer Check Protocol. Furthermore the BGP routing protocol engine must be extended to use the Adjacent Peer Check Interface in order to receive the events from the Adjacent Peer Check Service.

Looking at fig. 7 that depicts the finite state machine of BGP, only minor changes have to be taken. Every *KEEPALIVE* message can be replaced by an Adjacent Peer Check message. The difference is that the finite state machine gets a periodic feedback from the Adjacent Peer Check Service to satisfy the timers—i.e., the *Hold Timer*—of BGP. In order to keep the amount of protocol changes small, it is necessary to periodically notify the BGP routing engine about the liveness of its peer. With this kind of integration no BGP inherent properties have to be changed. But although the Adjacent Peer Check Service notifies the BGP routing engine in time intervals like e.g. every 90 seconds (to simulate *KEEPALIVE* packets), the Adjacent Peer Check Service is still able to react much more faster on link outages than the common BGP does. As soon as the Adjacent Peer Check Protocol detects the failure of a peer, i.e., a link outage or the malfunction of the routing control plane, an event towards the BGP routing engine is issued. This event causes the finite state machine of BGP to change its state from *ESTABLISHED* to *IDLE*, which tears down the BGP session with the broken peer.

B. Improvements

Using this adaptive Adjacent Peer Check approach could reduce [1] the amount of *UPDATE* messages that are issued due to physical line problems. The presented approach could take the properties of the physical line into consideration, which enables BGP to act more tolerantly on unstable lines. On the other hand it is also possible to detect a failure—whatever kind, i.e. protocol instance or link—on highly reliable physical lines much more faster than BGP does nowadays.

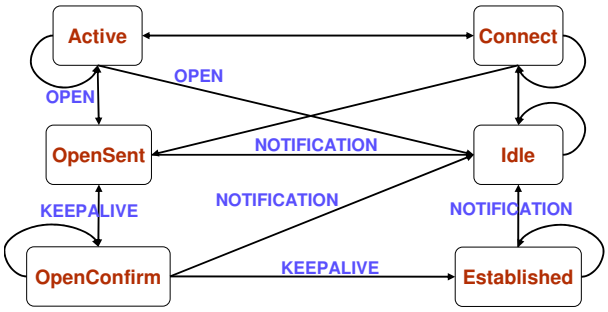

Fig. 7. BGP Finite State Machine

A further major improvement is that, which the reduction of updates, the Internet has a bigger chance to get to a coherent view of its topology, which makes the AS-paths more stable. The stability of AS-paths is very important for mission critical applications, where any down-time can hardly be tolerated.

C. Combining APCS with Fast Scoped Rerouting

The recently proposed Fast Scoped Rerouting (FaSRo) [13] extension for BGP enhances the Border Gateway Protocol to handle short time link failures, due to misconfiguration or physical link problems, in a local scope manner. For the FaSRo BGP extension it is very important that the link failure is detected immediately after the link outage occurs.

Combining those two approaches reduces the time until a failure is detected and allows to handle the failure in a local manner at the same time.

The Adjacent Peer Check Service does not only provide a fast failure detection time, but also reduces the amount of the so-called false positive detections of link or routing protocol instance failures. So, using the Adjacent Peer Check Service provides a contribution to the reduction of unnecessary BGP *UPDATE* messages.

VII. Conclusion

The failure detection mechanisms used in modern, widely-deployed routing protocols are not designed or configured for small failure detection times. This problem wastes precious time after a failure, since all reactions by the routing protocols are delayed unnecessarily. After looking at the used mechanisms and identifying the similarities, an architecture for extracting a general failure detection service has been developed, which can be used by any adapted routing protocol.

One of the main design goals of this novel Adjacent Peer Check Service has been the possibility to configure a very high frequency of check messages. This degree of freedom in configuration allows for a quick and efficient failure detection. The functionality has been evaluated in practical tests with an implemented prototype.

It has been shown, how popular routing protocols would have to be modified to take advantage from such a service with a special consideration of BGP. In a modular system like this, adapted routing protocols could immediately benefit from improvements of the failure detection service.

References

[1] R. Mahajan, D. Wetherall, and T. Anderson, Eds., *Understanding BGP Misconfiguration*, 2002, Sigcomm 2002.

[2] J. Kunzmann, G. Lichtwald, and U. Walter, “Entwurf und Implementierung eines externen Peer Checks für Routing Engines,” June 2003.

[3] K. Kompella, “Protocol Liveness Protocol,” IETF, Oct. 2002, draft-kompella-rag-plp-00.txt.

[4] C. Hedrick, “Routing Information Protocol,” RFC 1058, IETF, June 1988.

[5] J. Moy, “OSPF Version 2,” RFC 2328, IETF, Apr. 1998.

[6] T. L. Y. Rekhter, “A Border Gateway Protocol 4 (BGP-4),” RFC 1771, IETF, Mar. 1995.

[7] D. Katz and D. Ward, “Bidirectional Forwarding Detection,” Internet Draft IETF, August 2003, draft-katz-ward-bfd-01.txt.

[8] I. S. I. U. of Southern California, “INTERNET PROTOCOL,” September 1981, rfc791.txt.

[9] S. R. Sangli, Y. Rekhter, R. Fernando, J. G. Scudder, and E. Chen, “Graceful Restart Mechanism for BGP,” March 2004, draft-ietf-idr-restart-08.txt.

[10] D. Oran, “OSI IS-IS Intra-domain Routing Protocol,” IETF, Feb. 1990, RFC 1142.

[11] H. Y. C. Alaettinoglu, V. Jacobson, “Towards Milli-Second IGP Convergence,” IETF, Nov. 2000, draft-alaettinoglu-isis-convergence-00.txt.

[12] IETF, “Global Routing Operations minutes,” Jul 2003, http://www.ietf.org/proceedings/03jul/166.htm.

[13] R. Bless, G. Lichtwald, M. Schmidt, and M. Zitterbart, “Fast Scoped Rerouting for BGP,” in *International Conference on Networks*, no. 11, IEEE. IEEE, September 2003, pp. 25–30, 11th International Conference on Networks, 28.09. - 01.10. Sydney, Australia, ISBN 0-7803-7788-5.