Flexible Strategy Configuration for efficient operation of a Next Generation Network

Uwe Walter
and Martina Zitterbart
Institute of Telematics
University of Karlsruhe (TH), Germany
Email: {walter, zit}@tm.uka.de

Joachim Charzinski
Siemens AG
Munich, Germany
Email: joachim.charzinski@siemens.com

***Abstract*— In times of network convergence and increasingly challenging customer demands, automated tools can help operating Next Generation Networks more efficiently. Since operators have different ideas and requirements about the strategies they use for network operation and maintenance, automated solutions need to be very flexible. This paper will present a concept that allows for an easy and flexible configuration of the operational strategy carried out by a traffic and performance management node.**

I. Introduction

One of the currently most important trends in the telecommunication industry leads towards combining traffic of different service types onto a converged IP-based network. Given the increasing importance of a stable and high-performance network connectivity for customers, future networks must be able to meet very demanding requirements.

The joint research project KING[1] (Key components for the Internet of the Next Generation) aimed at the development of such a Next Generation Network (NGN) that features support for Quality of Service in combination with carrier-grade resilience [1], [2].

To enable a network provider to operate a KING network efficiently, a Network Control Server (NCS) has been developed. Its task is to monitor the network's operating conditions and adapt its parameters if necessary, e.g. after changing traffic patterns or topology changes like link failures or reconnections.

To influence the network's behavior, the NCS is equipped with algorithms that control the internal routing and Network Admission Control (NAC). The former is done by adjusting the link metrics and configuring the routers accordingly. Admission of high priority traffic is regulated by an admission control at the network border routers. Each NAC instance is able to administrate a given local traffic budget on its own to reduce signaling overhead and increase resilience. These traffic budgets are upper bounds on the maximum amount of admissible traffic at each NAC instance (and per service class), calculated and optimized to the current capacity and load situation by the Network Control Server.

This architecture allows to combine the advantages of a central management node and distributed functions. While the NCS is able to optimize network parameters to the current situation more efficiently than each network component on its own, care is taken not to make network availability depend on a working NCS. All real-time tasks, i.e., packet forwarding, QoS signalling and failure reaction, are handled autonomously by distributed network components (routers, NAC boxes). Thus, a temporary failure of the Network Control Server has no impact on the basic network operation. For maintaining QoS after link failures and re-routing, admission control budgets are used that admit only traffic that can still be carried after a failure.

In addition to keeping the network in a well-balanced operating condition, the NCS can relieve operators of routine maintenance tasks and aid in the more complex tasks, e.g. in traffic engineering for changing traffic matrices or in evaluating network upgrade options.

II. Steering Process of the NCS

Before deploying a Network Control Server, which is able to automatically control certain aspects of the network behavior by changing operational parameters (e.g. adjusting link weights to influence the internal routing), it is necessary to define a strategy that determines how to evaluate network conditions and how to react to changing conditions in order to achieve a controlled result.

Within the KING project, an NCS prototype was built with a steering process that follows the strict policy to only change network parameters if really necessary and useful. Furthermore, if a reaction is necessary, the possible influence on network operation is sought to be kept to a minimum. To help in a better understanding of the strategy examples given in the remainder of this paper, the basic reaction process of the NCS is shortly outlined in the following.

A. Basic reaction process

Since a network provider seeks to maximize the income generated by its network infrastructure, one of the most important tasks of Traffic and Performance Management is to

[1]Parts of this work were funded by the German Ministry of Education and Research (BMBF) under contract 01AK045. The authors alone are responsible for the content of this paper.

enable the network to transport the biggest possible amount of high priority traffic. Therefore, the probability of a request for high priority traffic getting blocked is one of the most important triggers for a necessary reaction, since such occurances directly correlate to lost income. The NCS continuously monitors the current blocking rates and budget utilizations by gathering statistical data from all NAC boxes to estimate the currently offered traffic matrix. Using this traffic matrix and the currently active NAC traffic budgets, the (non-linear) multi-rate Erlang blocking function transforms the estimated offered traffic rates into blocking estimates. This approach allows a good warning indicator for likely blocking, even before it actually occurs, whereas simply relying on measured blocking events would only allow a reaction after a significant number of requests were blocked.

Having computed the estimated blocking rate, the NCS compares this to configured thresholds. These represent the provider's notion of an acceptable blocking rate, which does not yet justify an intervention. If the blocking rate is above an acceptable level, the NCS uses its *Budget Assignment* algorithm to re-optimize the NAC budgets to the current traffic matrix [3], [4]. Having sent out the new budgets to the NAC instances, the blocking rate is evaluated again.

There is a good chance that re-optimization of traffic budgets alone is sufficiently successful in reducing the blocking rate. While offered traffic at one network border router might have increased, it might have diminished at another one. Since the Budget Assignment algorithm takes the network topology into account, it is often able to shift unused traffic budgets from one NAC instance to another, reducing the likelihood of blocked QoS requests at the higher loaded network entry point.

The KING Admission Control does not preempt data flows once they have been successfully admitted to the network. Thus, when reducing a budget below the currently booked level, all new requests will be blocked until enough flows terminate on their own, freeing enough new traffic budget. Together with the built-in resilience, this mechanism prevents budget changes from affecting admitted traffic.

On the other hand, the more powerful tool of changing the network's internal routing might result in a period of routing instabilities with increased packet loss rate during the convergence phase. This is why the NCS does only trigger its *Metric Optimization* algorithm [5] if blocking remains too high even after a re-optimization of NAC budgets. To conclude the efforts of saving the network unnecessary configuration changes, the NCS is able to evaluate the gain of a new set of link metrics in advance and can decide to discard them if there would be only marginal benefit from their configuration.

B. Failure reaction

Whereas the basic reaction process runs periodically to adapt the network to possibly slow-going changes of the traffic matrix, there are other triggers such as link failures where a faster reaction is advisable. The NCS monitors the message exchange of the internal routing protocol to get notified about topology changes.

As described in section I, due to resilience aspects, the Network Control Server must not be mandatory for the real-time failure reaction. This is why the optimization algorithms running on the NCS are prepared to take certain failure scenarios into account (e.g. each possible single link failure), allowing to calculate traffic budgets and link metrics that are already suitable for possible failures in advance. Routers continue to handle failures locally by immediately re-routing traffic according to their link metrics and topology information. On the other hand, a reaction of the NCS, e.g. by reducing traffic budgets, is not immediately due but can be delayed a little bit to benefit from the fact that most failing links are reconnected after only a few minutes [6], [7].

The failure reaction of the NCS incorporates a *Holdoff*-Timer: If the failure has already been anticipated during the calculation of the current traffic budgets, a reaction is delayed until the configured Hold-Timer expires. If this happens, the new topology is used as basis for a new optimization of network parameters (and prepare the network for possible next failures). However, if the failure gets repaired before the Hold-Timer expires, an unnecessary reaction was successfully avoided.

III. Flexible Strategy Configuration

While the basic strategy outlined in section II already allows to configure certain thresholds, timers and scenarios, like the acceptable blocking rate, the Holdoff-Timer and the anticipated failure scenarios, network operators might want to adapt the process to their needs in a more flexible way. One can expect a great variety of demands towards the configuration of an NCS strategy. While some network providers might be content with pre-defined basic strategy examples and only need minor tweaking to their policy, others might prefer to completely change the logic behind the default steering process and build their own complex reaction strategies.

To allow each operator to easily define their desired notion of the appropriate reaction strategy, the Network Control Server has been divided into functional building blocks, e.g. the different triggers, optimization algorithms and configuration tools. Network parameters and statistical data are represented by different data types, e.g. the current traffic matrix, link metrics, thresholds, etc.

An operator can use either a textual pseudo-code mode or a graphical representation, to combine these building blocks and elements into the desired reaction strategy.

A. Pseudo-Code Example

Operators with programming experience might prefer to define their reaction strategy in a simple pseudo-code representation, very similar to e.g. the Java programming language.

An example of such a pseudo-code representation is given in figure 1. This example[2] is implementing the strategy de-

[2]While space constraints do not allow the explanation and purpose of each data type and function, the example code should be intuitively understandable.

```
doPeriodically(120){
  // compute new ("active") offered traffic matrix
  tTM newActTM = fCompActiveTM(gGraph,
    fGetNACData(tNow-130s,tNow-10s),plannedTM);
  tBlockingData blockingTest = fCompBlocking(newActTM,
    gBudgets);
  if (fCompSelectMax(blockingTest) <= blockingLimits) {
    return;
  } else {
    tBudgets newBudgets = fCompBudgets(gGraph, newActTM,
      gMetrics, gFailScen);
    fActConfigureBudgets(newBudgets);
    gBudgets = newBudgets;
    tBlockingData blockingTest = fCompBlocking(newActTM,
      gBudgets);
    if (fCompSelectMax(blockingTest) <= blockingLimits) {
      return;
    } else {
      tMetrics testMetrics = fCompMetrics(gGraph,
        newActTM, gFailScen);
      tBudgets testBudgets = fCompBudgets(gGraph,
        newActTM, gMetrics, gFailScen);
      tBlockingData blockTestM = fCompBlocking(newActTM,
        gBudgets);
      if (fCompSelectMax(blockTestM) > blockingLimits) {
        if (fCompSelectMax(blockTestM) >
      cSignificanceFactor * blockingTest) {
          fMethAlarm("no significant improvement");
          return;
        } else {
          fMethInformOperator("metrics significantly
    improved blocking but failed to reach target");
        }
      }
      gMetrics = testMetrics;
      gBudgets = testBudgets;
      fActConfigureMetrics(gMetrics);
      fActConfigureBudgets(gBudgets);
    }
  }
}
```

Fig. 1. Excerpt of a pseudo-code strategy configuration example

scribed in subsection II-A, where the current blocking rate is periodically calculated and compared to a given threshold. If the blocking rate is too high, a budget reassignment is tried first, before new link metrics are calculated. If all options fail to reduce the blocking rate, warnings are sent to the operator, who might want to plan a network upgrade to react to the increasing amount of offered high priority traffic.

While the code example does show a periodic task, the reaction strategy is not limited to a sequential process. Triggers are modelled similar to interrupts or exceptions in common programming languages. For example, the expiration of a timer or a signalled change in the network topology (e.g. due to a link failure) could also be assigned with appropriate reaction code. A prioritization system with higher execution priority for urgent reactions would be used to avoid multitasking conflicts.

B. Graphical Strategy Definition Example

While many operators might be comfortable with a pseudo-code like strategy description, some might prefer a more visual configuration. The latter could take advantage of a graphical representation of the strategy definition.

Figure 2 depicts such an example, implementing the same strategy as the former pseudo-code example. A graphical user interface (GUI) would help an operator to intuitively build such a strategy definition out of the available functional building blocks by interconnecting them (e.g. via drag-and-drop actions).

To reduce the complexity of the representation, it would be possible to define own macro blocks, resulting in a module hierarchy. If some of the usually sequential tasks of figure 2 are combined into reasonable macro blocks, this would result in the less cluttered version shown in figure 3.

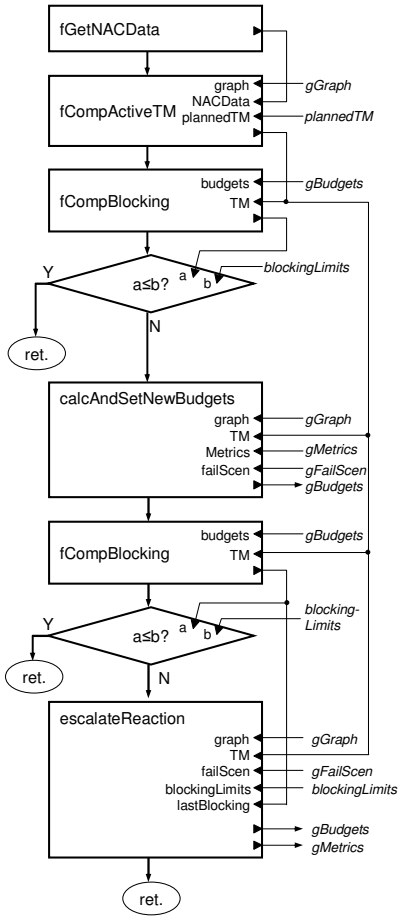

Fig. 3. Example from figure 2 with some defined macro blocks

C. Further Benefits

A more flexible way of configuring a system usually increases the possibility for misconfigurations. Therefore, appropriate mechanisms should be built into the tools responsible for the pseudo-code or GUI like definition. These should help to disallow (or at least detect) configuration errors, like data type mismatches, contradicting combinations of building blocks, endless loops and more (of course limited by the possibilities of automated semantic checks). In addition, predefined procedures or macro blocks should be delivered to allow the easy construction of default strategies.

If the full potential of the described strategy configuration would be used, operators could greatly benefit from the possibilities. E.g. it would be possible to change the quality of the optimization results by using different numbers of optimization steps, depending on the urgency of the new values (many links

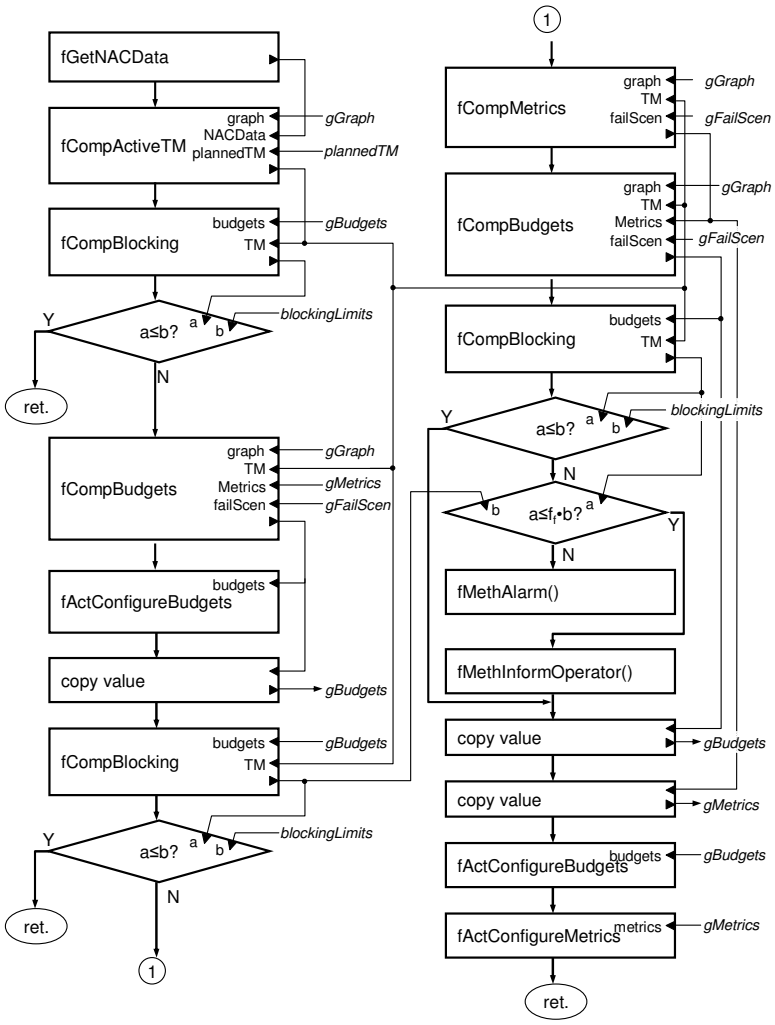

Fig. 2. Example of a graphical strategy definition

failing at once make a more imminent reaction necessary than a single failure).

The time of day or the current link loads could influence the strategy decision. Maintenance tasks or more complex network reconfigurations could automatically be performed if certain network statistics (e.g. traffic loads) would fall below given thresholds.

IV. Conclusion

This short paper has outlined the concept of a highly flexible strategy configuration for a control server to be used in a Next Generation Network. Using the presented framework, network operators can quickly define their desired operating strategy and adapt it to changes of their policies. Automated assistance during the setup process helps reduce configuration errors which might otherwise negatively impact the customers' network experience.

In addition to keeping the network in an optimized operating condition, the benefits of the flexibly configurable Network Control Server help to efficiently operate a NGN, which is one of the key requirements for Network Management today.

References

[1] C. Hoogendoorn, J. Charzinski, K. Schrodi, N. Heldt, M. Huber, C. Winkler, and J. Riedl, "Towards the Next Generation Network," in *12th IEEE International Conference on Network Protocols (ICNP 2004), Berlin, Germany*, Oct. 2004.

[2] K. Schrodi, "High Speed Networks for Carriers," in *7th IFIP/IEEE International Workshop, Protocols for High Speed Networks (PfHSN 2002), Berlin, Germany*, Apr. 2002.

[3] M. Menth, S. Kopf, and J. Charzinski, "Network Admission Control for Fault-Tolerant QoS Provisioning," in *7th IEEE International Conference on High Speed Networks and Multimedia Communications (HSNMC)*, Toulouse, France, June 2004, pp. 1 – 13.

[4] M. Menth, "Efficient Admission Control and Routing in Resilient Communication Networks," PhD thesis, University of Würzburg, Faculty of Computer Science, Am Hubland, July 2004.

[5] C. Reichert and T. Magedanz, "A Fast Heuristic for Genetic Algorithms in Link Weight Optimization," in *5th International Workshop on Quality of Future Internet Services (QoFIS), Barcelona, Spain*, Sept. 2004.

[6] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP Misconfiguration," in *Sigcomm, Pittsburgh, USA*, Aug. 2002.

[7] O. Bonaventure, C. Filsfils, and P. Francois, "Achieving Sub-50 Milliseconds Recovery Upon BGP Peering Link Failures," in *First International Conference on Emerging Networking Experiments and Technologies (CoNEXT), Toulouse, France*, Oct. 2005.