 The definitive version of this paper has been published in Tenth IFIP-IEEE International Symposium on Integrated Network Management (IM 2007), 2007, 10.1109\/inm.2007.374721.

An Autonomic Approach to Verify End-to-End Communication Quality

Andreas Binzenhöfer, Daniel Schlosser, Kurt Tutschku
University of Würzburg
Institute of Computer Science
Würzburg, Germany
[binzenhoefer,schlosser,tutschku]@informatik.uni-wuerzburg.de

Markus Fiedler
School of Engineering
Blekinge Institute of Technology
Karlskrona, Sweden.
markus.fiedler@bth.se

***Abstract*— The complexity of today's computer networks requires highly trained professionals to verify the end-to-end connectivity between two communication partners. The concept of autonomic networks has been proposed in an effort to make such networks easier to manage. In this context, we present a plug-and-play mechanism to check the connectivity between two end points of the network. It will significantly decrease the amount of mundane work, which is usually involved in such a task. The main goals of our work are to autonomically set-up the measurement environment, to capture the current conditions between the two communication partners and to visualize the results in an easy-to-interpret way.**

I. Introduction

One of the major problems of today's distributed applications is to find the root cause in case of a failure or decreased functionality. This is especially difficult since there are multiple areas of responsibility involved. Usually the blame is shifted from application level via other levels like the operating system all the way down to the network itself. On account of this a considerable part of the work of network administrators consists in proving that it is not the fault of the network. This creates an urgent need (1) *to monitor the user-perceived quality* and the *end-to-end connectivity across multiple networks*. It is also non-trivial to (2) *set-up and maintain the end-to-end measurement*, especially in the face of mobile users, dynamic IP-addresses, and Network Address Translation (NAT). Another question is which parameters to measure and how to obtain the measurements? Finally, (3) *the measured values must be interpreted* in order to derive a meaningful statement.

In this paper we approach the three above mentioned problems by introducing an easy-to-use concept, which supports the administrator in verifying the quality of end-to-end communications in a more autonomic way. It uses passive measurements to derive network parameters like jitter, packet loss, or one way delays. These values are then mapped to a traffic light scheme which can intuitively be understood without the need of a well trained professional. The results can be used to indicate whether the network supports a specific service like a VoIP call, a videoconference, or file transfers. To prove the applicability of our approach we integrate the concept into our autonomic distributed network management prototype [1] and perform a case study. Related work is referred to in the corresponding sections.

II. Description of the Architecture

The main architecture consists of two parts. An overlay network which provides the plug-and-play framework for distributed network management [1] and the modules which offer the measurement functionality.

A. Network Management Overlay

It was previously suggested to use p2p networks for distributed network management [2]. While most approaches concentrate on a specific monitoring task, we presented a generic framework for distributed network management [1]. It consists of our distributed network application (DNA) which runs on several end-hosts and enables the peers to communicate with each other independent of their current location, their current IP address, or the kind of device they are using. The algorithm used to organize the overlay network is based on a modified version of the Kademlia protocol as described in [1]. Kademlia [3] is a structured p2p network which can guarantee to efficiently locate any online user.

B. Verification of the End-to-end Connectivity

The implementation is realized using three different modules which are integrated into our DNA client as shown in Fig. 1. Module M1 checks the local configuration and

Fig. 1. The modules providing the measurement functionality

determines whether the device is ready to use the network. M2 measures the characteristics of the traffic sent into the network and exchanges this information with the corresponding module of the communication partner. The results are mapped to a simple traffic light scheme, which is easy to understand by any end-user. M3 represents a traffic generator, which is able to emulate specific services like Skype calls or traffic with a constant bitrate.

1) M1: Ready for Use: The first step in verifying the end-to-end connectivity is to exclude errors which actually have a local cause. The idea is to automate all the things an administrator usually does when he checks the local configuration of an end host. In terms of autonomic networking it also greatly reduces the amount of mundane work required by a human network specialist. In fact, there are a lot of routine tests, like checking the IP configuration, which can be automated. A detailed list can be found in the technical report [4]

2) M2: Traffic Light: The traffic light module measures the network traffic on two specific end-host, exchanges information about the measurements and analyzes the effects of the network on the user perceived quality. The results will be visualized in a traffic light scheme. The quality can either be good (green), acceptable (yellow), or bad (red). There are only two things which need to be configured before the module can start with the measurements: The overlay name of the remote host and R_T, the time between the exchange of two results (default 10s). The interval R_T indirectly determines how frequently the color of the traffic light is updated.

Using the search functionality of the overlay network, the traffic light module is able to automatically determine the IP-address of the communication partner by its overlay nickname. While the module is running, it captures all packets which match this IP-address and an optional port. For each of these packets it records the IP-ID, the timestamp, and the size of the packet. Based on all outgoing packets captured within a measurement interval host B creates a result summary as shown in Table I and sends it back to host A.

TABLE I
STRUCTURE OF THE SUMMARY PACKET

Name	Size	Description
SP_S	1 bit	0 for a summary packet, 1 for a summary packet response
C	2 bit	Color of downlink traffic light of host B
encoding	1 bit	0 if $\Delta t^B_{out}(i)$ are 16 bit integers, 1 if $\Delta t^B_{out}(i)$ are 32 bit integers
S_{ID}	4 bit	summary packet ID
$t^B_{out}(1)$	8 byte	time stamp of first outgoing packet
$\Delta t^B_{out}(i)$	2/4 byte	times between outgoing packets i and $i-1$, if $i > 1$, 0 otherwise
$ID^B_{out}(i)$	2 byte	IP-ID of packet i

The first bit SP_S determines whether the packet is a summary packet or an acknowledgment. The remaining information in the packet suffices to recreate the timestamps $t^B_{out}(i)$ and the IDs $ID^B_{out}(i)$ of all outgoing packets of host B. Host A will then compare these values to the corresponding timestamps $t^A_{in}(i)$ and $ID^A_{in}(i)$ to derive the color of its downlink traffic light. To protect the exchange of the summary packet against packet loss, host A sends an acknowledgment message to host B. This message includes the median of the measured one way delays of link AB m_{AB}, the jitter, and the throughput.

3) M3: Traffic Generator: In many situations it is interesting to know whether an existing network is suitable for a new application. In this case a passive measurement approach would require to actually install the application before the measurements can be done. To overcome this problem, a traffic generator is added to the traffic light module. It is initialized with the distribution of the interarrival time of the packets and the distribution of the payload. It is also possible to use other predefined random streams or to replay captured traffic streams.

III. Analyzing and Visualizing the Measurements

In this section we will show how to derive one way packet loss, jitter, and delay and how to map those values to a traffic light scheme for different services. Without loss of generality we will concentrate on host A. It is able to capture the IDs $ID^A_{in}(i)$ and the timestamps $t^A_{in}(i)$ of all incoming packets according to its own clock. From the summary packet sent by host B it also knows the IDs $ID^B_{out}(i)$ and the timestamps $t^B_{out}(i)$ according to the clock of host B.

A. One Way Packet Loss

The calculation of the packet loss is straightforward. Host A simply uses the packet IDs to determine which of the packets sent by host B never arrived at host A. That is, it regards all incoming packet IDs $ID^A_{in}(j)$ with

$$ID^B_{out}(1) \leq ID^A_{in}(j) \leq ID^B_{out}(i_{max})$$

and calculates the percentage of missing IDs. In the following we will assume that $i = 1, ..., i_{max}$ is the index of the ith successfully transmitted packet from host B to host A.

B. One Way Jitter

In order to calculate an estimate for the one way jitter over time, we draw on the formula applied in the real-time transport protocol (RTP) [5]. From the time stamps in the summary packet we calculate $\Delta t^B_{out}(i)$, the time between outgoing packet i and $i+1$ at host B, and $\Delta t^A_{in}(i)$, the interarrival time of packet i and $i+1$ at host A. We initialize the jitter on link BA with $j_{BA}(1) = 0$ and estimate the current jitter as

$$j_{BA}(i) = j_{BA}(i-1) + \frac{\left|\Delta t^B_{out}(i) - \Delta t^A_{in}(i)\right| - j_{BA}(i-1)}{16}$$

The gain parameter is set to $\frac{1}{16}$ since this gives a good noise reduction ratio while maintaining a reasonable rate of convergence [5].

C. One Way Delay

An accurate time synchronization is absolutely essential for the measurement of the one way delay. There are different ways to achieve synchronized clocks. The most prominent examples are atomic clocks, GPS, and the network time protocol (NTP) [6]. Let Θ be the difference in milliseconds between the clocks of host A and host B. Without loss of generality we assume that $Clock_A = Clock_B - \Theta$. If $D_{BA}(i)$ is the measured delay of the ith successfully transmitted packet from host B to host A, then the real delay would be $D_{BA}(i) + \Theta$. For unsynchronized clocks, we have to assume that Θ might be quite large. In this case, one usually has to measure the round trip delay, divide it by two, and use the outcome as an estimate for the one way delay.

In general, we do not know whether the clocks of host A and host B are synchronized or not. Therefore host A initially calculates the one way delay of the ith packet on link BA as $D_{BA}(i) = t^A_{in}(i) - t^B_{out}(i)$. As long as all delays in both directions are positive, Θ has to be smaller than the shortest actual one way delay. In this case we assume sufficiently synchronized clocks and take $D_{BA}(i)$ as a direct estimate for the one way delay on the downlink of host A. In the worst case this estimate is twice as large as the actual delay.

If otherwise one of the measured delays is negative, we assume unsynchronized clocks and $\Theta >> D_{BA}(i)$. However, instead of performing additional active round trip time measurements, we can easily adjust our passive results. Each host calculates the median m_{BA} (or m_{AB} respectively) of the measured one way delays on its side and inserts it into the acknowledgment of the summary packet. If one of these values is negative, it will recalculate its estimates of the one way delays. In case $m_{BA} < 0$ this results in

$$D'_{BA}(i) = D_{BA}(i) + \frac{m_{AB} - m_{BA}}{2}.$$

The equation assumes symmetric one way delays and readjusts the measured values in such a way, that the median one way delay is equal in the uplink and the downlink.

Another problem, which has not yet been discussed is that computer clocks tend to tick at different rates. Therefore, we estimate the clock drift d_{rel} according to [7] and remove the drift by adjusting the one way delays as follows:

$$D'_{BA}(i) = D_{BA}(i) - d_{rel} \cdot \left(t^A_{in}(i) - t^A_{in}(1)\right).$$

D. Calculating the Traffic Light

Different services react differently to changes in the network. While real time services have strict delay and packet loss requirements, the quality of a file transfer mainly depends on the average throughput. Utility functions [8] are a good way to translate network dynamics into humanly understandable terms. They take parameters like packet loss, jitter, and delay as input and deliver a number between zero and one, which reflects the current user perceived quality.

In the current version of our prototype we decided to use simple tables to determine the color of the traffic light. For VoIP connections, e.g., we defined Table II according to the ITU-T recommendation G.114 [9].

TABLE II
TRAFFIC LIGHT COLORS FOR VOIP

	delay [ms]	packet loss [%]	jitter [ms]
green	<75	<1	<10
yellow	<200	<3	<25
red	>200	>3	>25

A more complex calculation of the traffic light color is part of our future work and is beyond the scope of this paper. However, there are numerous algorithms and models for real time services which can also directly be integrated into our prototype. The E-model [10], e.g., is a well established computational model that uses measured network parameters to predict the subjective quality of voice calls which can then easily be mapped to our three-color scheme. Another interesting approach to quantify user satisfaction for voice applications is shown in [11], where the authors derive a user satisfaction index based on bit rate, jitter and round trip times. Finally, in [12] a set of measurements that can be used to derive a media delivery index is described in detail.

IV. PROTOTYPE STUDY AND RESULTS

To validate the feasibility and the practicability of our approach we implemented the traffic light concept as an extension of our DNA framework [1], [4] and set up a small testbed environment in our laboratory. As shown in Figure 2 it consists of two end hosts A and B running Windows XP and a Linux server routing the traffic between the two end hosts. Our prototype was installed on host A and host B and we use NISTNet [13] on the router to emulate one way delays, packet loss, jitter, or asymmetric bandwidths. We used

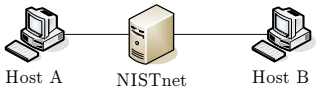

Fig. 2. Setup of the testbed

100Mbit/s links and a 30ms one way delay in both directions and monitored a Skype VoIP call which periodically sends packets every 30ms. The prototype was able to detect the unsynchronized clocks as well as the clock drift and adjusted the one way delays according to Section III-C.

To study the influence of the time between two summary packets, we have a closer look at the measured packet loss. We increase the current packet loss by five percent every 30 seconds until it reaches 20 percent, then we decrease it again by five percent every 30 seconds until it is back to zero percent. After 60 seconds we repeat the entire procedure. Figure 3 shows the packet loss as observed by the prototype when sending a summary packet every 2, 10, or 30 seconds, respectively. Since packet loss occurred randomly and uncorrelated a measurement interval of 2 seconds leads to heavily fluctuating values. While a measurement interval of 30 seconds quite accurately reflects the current loss rate, it reacts too slowly to changes in the network. That is, the end user would have to wait for 30 seconds until the traffic light would adapt. We therefore chose an interval of 10 seconds as a good trade-off between accuracy and up-to-dateness.

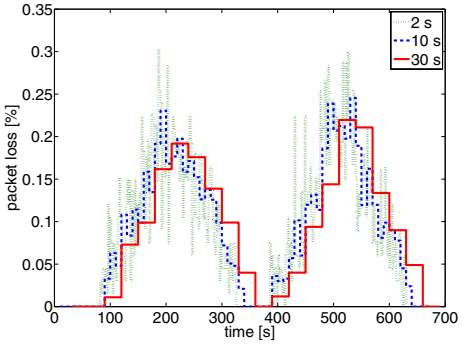

Fig. 3. Loss at different resolutions

In the following we again use a VoIP call to determine how the measured network parameters translate into user-perceived

quality. Figure 4 shows the measured one way delays on the left y-axis and the current color of the traffic light on the right y-axis. We started with a plain connection between host A and host B, translating into a green color. After 60 seconds we increased the one way delay to 50ms, which did not change the color of the traffic light. After another 60 seconds we increased the delay to 150 seconds with a jitter of 30ms and introduced an additional packet loss of 3 percent. After a slight delay the color of the traffic light changed to yellow. When we further increased the delay to 300ms with 40ms of jitter and 10 percent packet loss, the traffic light finally changed to red, signaling a bad quality. After 60 seconds we decreased the delay to 100ms with 10ms of jitter and the packet loss to 2 percent, which was reflected by a yellow traffic light. Finally, we returned to 50ms delay without any jitter or packet loss and observed a green traffic light.

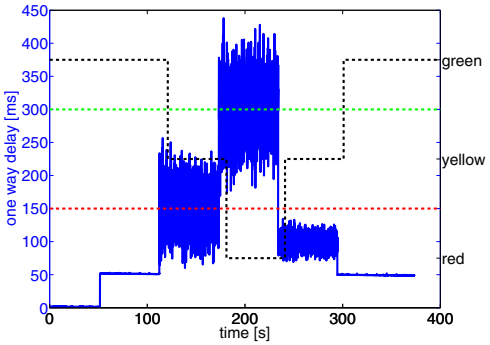

Fig. 4. Traffic light and delays

To show that the color of the traffic light of our prototype actually reflects user perceived quality, we recorded the voice quality on application level at host B and compared it to the original quality as sent by host A. To do so, we periodically calculated the "Perceptual Evaluation Of Speech Quality" (PESQ) value [14] of a typical 10 second conversation [15], which was repeatedly transmitted with a 2 second pause between transmissions. A PESQ value above 3 corresponds to good quality, while it is still considered to be acceptable between values of 2 and 3.

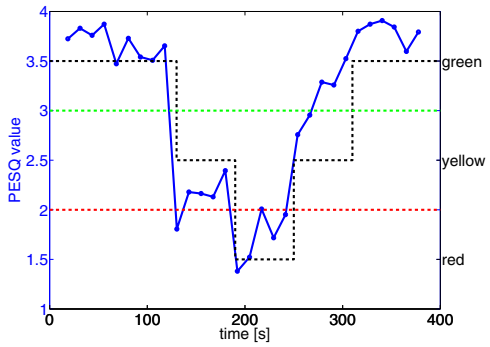

Fig. 5. Traffic light and PESQ

Figure 5 shows the current PESQ value on the left y-axis and the color of the traffic light on the right y-axis for the previously described example. The color of the traffic light corresponds to the trend of the PESQ values. It stays green while the PESQ value is above 3.5 and changes to yellow once the PESQ value decreases to values between 2 and 2.5. During the phase in which the traffic light shows a red color, the PESQ value falls as low as 1.4. Note that the PESQ value constantly rises during the following yellow phase, while the network settings rather imply steady values. The reason is that Skype actually adapted its voice codec to the current condition of the network (c.f. [4]).

V. Conclusion

The quality of a service as it is perceived by the end-user is an aspect which is often neglected in classic network management. In this paper we presented a self-organizing approach to evaluate the quality of specific end-to-end communications. It measures important network parameters like the one-way-delay, jitter, and packet loss on both endpoints of the communication. Correlating these traffic characteristics of both communication partners, it translates the measured values to an easy to understand traffic light scheme.

As a proof-of-concept, the algorithm was integrated as an additional module into our already existing distributed network management environment [1]. The feasibility of this concept was verified by monitoring a Skype VoIP call in a small testbed. It could be shown that the color of the traffic light corresponded to the current PESQ value. That is, the traffic light reflected the subjective impression of the voice quality. The results can be used by network administrators to inspect the current state of their network, by service providers to verify that they preserve a negotiated service level agreement, or by (mobile) users to see whether the quality of their current connection allows the use of a specific service.

References

[1] A. Binzenhöfer, K. Tutschku, B. a. d. Graben, M. Fiedler, and P. Arlos. A p2p-based framework for distributed network management. In *New Trends in Network Architectures and Services, LNCS 3883*, Italy, 2006.

[2] Bugra Gedik and Ling Liu. A scalable p2p architecture for distributed information monitoring applications. *IEEE Trans. Comput.*, 54(6), 2005.

[3] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS*, Cambridge, USA, 2002.

[4] A. Binzenhöfer, D. Schlosser, and B. a. d. Graben. A novel approach to verify end-to-end connectivity in an autonomic way. Technical Report 383, University of Wuerzburg, 2006.

[5] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, 1996. RFC 1889.

[6] David L. Mills. A brief history of NTP time: memoirs of an Internet timekeeper. *SIGCOMM Comput. Commun. Rev.*, 33(2):9–21, 2003.

[7] Vern E. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD dissertation, University of California, 1997.

[8] M. Fiedler, S. Chevul, O. Radtke, K. Tutschku, , and A. Binzenhöfer. The network utility function: A practicable concept for assessing network impact on distributed services. In *ITC19*, Beijing, China, 2005.

[9] ITU-T Recommendation G.114. One way transmission time, May 2003.

[10] ITU-T Recommendation G.107. The e-model, a computational model for use in transmission planning, December 1998.

[11] K. Chen, C. Huang, P. Huang, and C. Lei. Quantifying Skype user satisfaction. In *ACM SIGCOMM*, Pisa, Itlay, 2006.

[12] J. Welch and J. Clark. A proposed media delivery index. Internet Draft, August 2005.

[13] M. Carson and D. Santay. Nist net: A linux-based network emulation tool. *ACM SIGCOMM Computer Commununications Review*, 33, 2003.

[14] ITU-T Recommendation P.862. Perceptual evaluation of speech quality (PESQ), an objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs, 2001.

[15] Signalogic. Speech Codec Wav Samples, http://www.signalogic.com/index.pl?page=codec_samples.