An Active Network Measurement Service for TAPAS Group Communication

Michael Dales[1], Antonio Di Ferdinando[2], Paul Ezhilchelvan[2] and Jon Crowcroft[1]

Abstract: This document outlines the network measurement layer designed to support the QoS Group Communication system. Together with deliverable D8 - *QoS-Enabled Group Communication* provides a full description of the TAPAS Group Communication facility.
This document is structured so as to discuss the system's requirements first, after which it provides a brief high level look at how the system is designed and concludes with a few words on the integration issues with the rest of the Group Communication system.

Introduction

In deliverable D8 (*QoS-Enabled Group Communication*) [diferdinandoD8] we introduced the Group Communication (GC) system designed for the TAPAS project. This system provides users with probabilistic delay guarantees over the entire group. In order to be able to provide this guarantee, the communication service needs to be aware of the current state of the network behaviour for the group, as described in [diferdinando04], in order to be able to carry out access control. The group communication system will be then capable of deciding whether accept or not a communication request, with a given delay and probability bounds, only if it knows whether the network conditions will be able to support the service level requested.
This document describes the Network Measurement System (NMS) designed to support the TAPAS GC protocol. It extends the TAPAS deliverable D8 so as to complete description of the GC System. Section 2 describes both functional and non-functional requirements for the NMS. Design is described in Section 3, whilst integration with the rest of the GC system is described in Section 4.

Requirements

In our system a group is defined as a set of processes engaged in a coordinated activity, whose size of the group is assumed to be known. Each couple of members is connected through a *channel*, liable of failure such as delays and losses. As a consequence, each member knows how to reach each other member.

[1] *Computer Laboratory, University of Cambridge, CB3 0FD, UK.*
[2] *School of Computing Science, University of Newcastle, NE1 7RU, UK.*

Functional requirements: In order to reason about QoS guarantees over a multicast group, the GC system needs up-to-date measurements for four particular network properties, and it is the NMS that must provide this information to the GC. However, it is easy to understand from Figure 1 that there are multiple paths involved in the network, and it is likely each link will have a different set of properties.

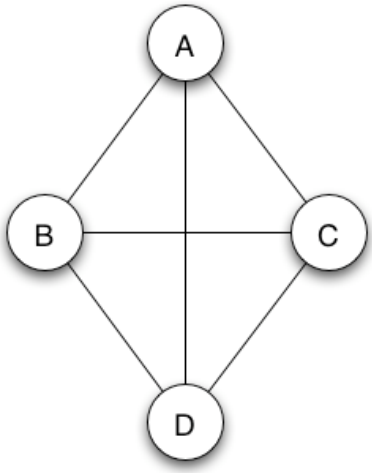

Figure 1 Simple 4 nodes network

For the group communication algorithm to work, the GC system needs to know the worse case value for each property. Given the multiplicity of links involved, then, the value to report back will be chosen as the maximum amongst the values for all paths. For example, each link will have a different average delay, and the NMS must report back the average delay for the link where this reaches the maximum value. A further complication arises if we consider that each different link could experience worst value: one link may be fast but exhibit high packet loss, whereas another link will be slow but have low packet loss. Thus, the worse value for each metric may actually come from a different link. Another important observation is that not all members may see the worse case link for a given metric. For example, node A in Figure 1 cannot see the network link between nodes B and C, yet this link may have the worse average delay, yet node A needs to be aware of this. All these observations lead the need for each individual node to share any network monitoring observations with all other members amongst the group. The four pieces of information required are:

- The average delay of the slowest link between any two nodes in the group. This gives a worse case packet delay for any message passed between two nodes in the group. (d)
- The delay distribution for the link with the worse average packet delay. This will describe the curve of the delay distribution, for example, uniform, exponential, etc.
- The average packet loss of the most error prone link between any two nodes in the group. This gives a worse case packet loss prediction for any message passed between two nodes in the group. (q)
- The average difference in transmission delays (i.e., the average *jitter*).

With all this information, the group communication layer can make the appropriate decisions for admitting communications. From this we can draw up the basic data flow between the group communication system and the network monitoring system, as shown in Figure 2. The group communication layer will provide the monitor with a list of group members, with which it may coordinate, and the network monitoring layer will provide the statistics required.

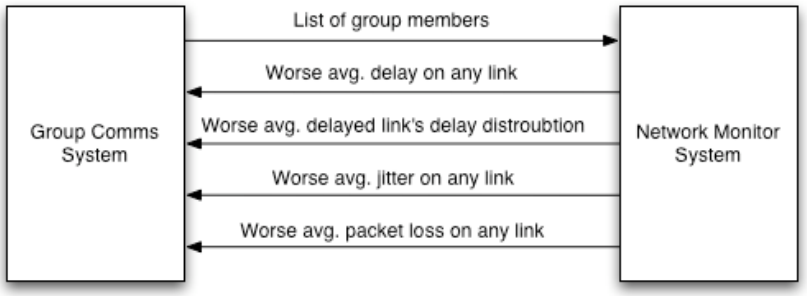

Figure 2 Data flow between GCS and NMS

Non-functional requirements: To interoperate with the already implemented group communication layer, the NMS must provide a *Java* interface. This interface will be callable by a Java program, which will provide a list of addresses of group members. Once started, the network measurement layer will report back periodically the state of the network using the four variables required.
The interface will be basic and will incorporate methods to monitor the network and update the variables. It could be as simple as

```
public interface Monitor{
    public void monitor();
    public void update();
}
```

Related work

Attempting to monitoring networks is not a new issue, and the system proposed here can take advantage of previous work done in this field. This section gives a quick overview of some relevant work, looking both at research in monitoring and the advice given by the Internet Engineering Task Force (IEFT) work group on IP Performance Monitoring (IPPM).
Given the commonality of network monitoring, the IEFT has a monitoring working group, IPPM, that is attempting to define standards for monitoring various network properties over a series of RFCs. RFC 2330[rfc2330] provides a set of standard definitions for monitoring, discussing the problems with trying to gain accurate measurements. One relevant piece of advice is on how to sample data. An obvious way to gather measurement data is through periodic sampling with a fixed interval, particularly given the simplicity of implementation. However, this can lead to unwanted side effects. Firstly, the measurement may happen to coincide with other regular traffic on the link, causing the measurement not to reflect the average performance on the network, but the performance at this interval. In addition, regular injection of measurement traffic in the network may cause a level of synchronisation of traffic on the network, causing the measurement to have a big effect on what is being monitored[floyd94].
The solution to these problems is to use a random distribution process, whereby the samples are taken at intervals with a random distribution around the ideal interval. The

example given in RFC 2330 is to use Poisson sampling intervals, which are simple to generate, and give the desired effect of removing the fixed periodic nature of the sampling.

The requirements for the network monitoring layer being built is that it measures the average delay between two nodes, commonly referred to as the One Way Delay (OWD). RFC 2679[rfc2679] provides the IPPM's definition of how to monitor OWD. As pointed out in RFC 2330, one important problem when trying to monitor networks is the lack of a global clock for making time comparisons. Getting two computers to agree on time is a very hard problem; even if it is possible to get the two computers to agree on the time at a single point, their internal clocks will not run at exactly the same rate, and they will drift apart over time. A common cheap solution to this problem is to use the Network Time Protocol (NTP) to repeatedly synchronise a computer's clock to a very accurate clock (e.g., an atomic clock). Periodically NTP will note the time as dictated by the more accurate clock and then slowly apply a series of small changes to the local clock until it is in line with the accurate clock (simply resetting the clock to the accurate time can potentially cause a large jump in time, which may have bad side-effects, and thus is not advisable).

This situation means that at any given time it is highly unlikely that any two computers will agree what the time is, so measuring OWD becomes a problem. OWD measurements using NTP for clock synchronisation has been shown to work under certain conditions[smotlacha02], but that required the hosts to have specific close relationships with NTP servers, and that cannot always be guaranteed. There have been attempts to use software to try to detect and counteract clock drift and NTP effects[moon99,zhang02], which monitor the skew effects and attempt to remove them. Another alternative is to use more accurate (and thus expensive) clocks at the end systems, such as a GPS based clock. However, that level of technology and cost is incompatible with the aim of the project, which aims to use software alone.

The alternative is to measure the Round Trip Time taken to send a packet from one computer to another and back again, which is covered by the IPPM in RFC 2681[rfc2681]. This technique only uses the clock of the sender to make a time comparison, and can thus does not require two synchronised clocks. Once a Round Trip Time (RTT) value has been produced, it can then be halved to generate an approximation for one way delay. It is important to note that this technique also has a problem, which is that it is not necessary for packets to traverse the same set of links in each direction, and thus the delays in each direction may not be equal.

Design

This section describes design for the NMS. Its architecture has been designed with all functional and non-functional requirements described in Section 2 in mind.

Local Link Assessment: The basic approach is that each node's network monitor will use active monitoring based on RTT measurements to determine the state of the link between itself and each of its neighbours. Using RTT measurements avoids the clock synchronisation issues outlined in Section 2.

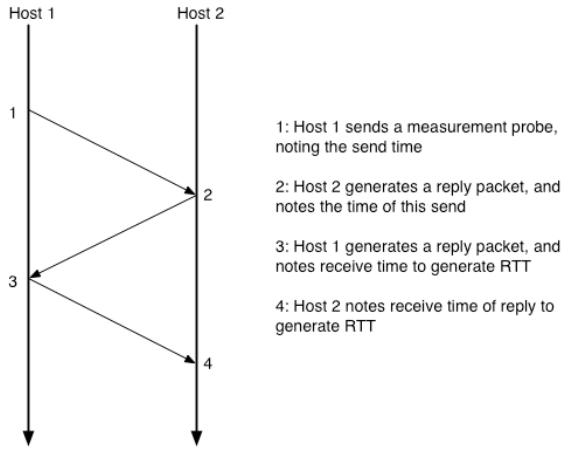

Figure 3 Overview of the three way ping algorithm

Periodically, based on a random distribution (as recommended by [rfc2330]), a monitor will attempt to measure one of the links between itself and another node. We do not attempt to measure all links at once, as that would potentially have a bad effect on the results as the monitor needs to process all the replies at a similar time, which would cause errors in the delay measurement. Each network monitor will use what we have termed a *z-ping* to generate monitoring information, as shown in **Error! Reference source not found.**. This attempts to reduce the load of measurement slightly by combining two RTT measurements into one, which requires three packets rather than four. In a single z-ping transaction both ends will be able to infer properties about the network rather than just the initiating node, which will mean that the average interval used between monitoring attempts by a node can be reduced, as often it will be able to take measurements based on another nodes attempts at measurement.

Should the RTT measurement be successful, within a certain timeout period, then the RTT value can be halved to get a delay value for that sample, which can then be used to calculate an updated average delay, delay distribution, and average delay variance for the link involved. If not, then the attempt is taken to be an indicator of packet loss

Estimating the Delay Distribution: As part of the working out the average delay, the network monitor will be required to hold a set of delay values over the last period of interest. These same values will then need to be used to work out the distribution of the delay values. The technique chosen for this is based on the *chi-square goodness-of-fit* test[james03]. The sample data will be divided into a series of buckets, and then the samples in these buckets is compared with how the buckets should be filled for a set of known distribution curves, with the curve parameters based on the measured mean and distribution. The values of the sampled data are then compared to the ideal curves using a the chi-square goodness-of-fit test which will indicate how well the data matches a given curve for a certain probability of accuracy.

This technique does require that we know before hand what type of curves we except to have to fit too, but that is also a requirement of the GC work, so this is acceptable.

Result Distribution: Over time the NMS at a particular node will build up a picture of the performance of the links between it and the other nodes in the group. However, the node needs to know the performance of the worse case links of the group overall. To do this, the nodes must periodically broadcast their results to the rest of the group. This will be done at a time scale longer than that of individual sampling, as the average performance is not expected to change as quickly as the instantaneous conditions on the

network will. Broadcasts could also be sent under special circumstances, such when the local conditions are noted to exceed a certain threshold. The message sent during results distribution contains the following four values:

- The worse delay average for any of the links seen by the node.
- The distribution of the delay average for the above link.
- The worse case delay variance for any of the links seen by the node.
- The worse case packet loss for any of the links seen by the node.

Upon receiving a broadcast, the node will look at all the 4-tuples of data it has from all the active nodes on the network, and then it will pick the overall worse values, and it is these values that will be reported up to the GC layer, as they represent the worse case link within the group, and thus the values that all nodes must use when making their probabilistic delivery calculations for access control.

Integration with the rest of the system

The GC system's components realize tasks that must be executed at different timings. Timely and sequential execution of them is vital for the protocol to correctly deliver the service. For this reason the user does not have direct access to the protocol. Its logic is encapsulated in a `Protocol` object that the user accesses. The NMC is instantiated by the `Protocol` object upon its own instantiation. This allows to have some data by the time the negotiation will have to take place.
In order to work correctly, it needs the list of members to monitor. Such list is passed in instantiation phase, and a dynamic group management protocol ensures the list to be updated, and consistent with the list of every other group member.
After instantiation, the NMS starts its tasks. The only component with which it communicates is the *Negotiation* component, to which it periodically passes generated network metrics in order to be used in the approximation process. The needed data is encapsulated into a custom data type, `NetData`, and passed to the *Negotiation* to be used. Communication between the two components is done by means of the use of a synchronized queue, where the NMS deposits a value that the *Negotiation* Component pops. In formation contained into the `NetData` is stored into *Negotiation*'s synchronized variables and used for approximation purposes.

Bibliography

[Cristian96] F. Cristian, "Synchronous and Asynchronous Group Communication", *Communications of the ACM*, 39(4):88-97, April 1996.

[rfc2679] G. Almes, S. Kalidindi, and M. Zekauskas. "A One-way Delay Metric for IPPM". *RFC 2679*, Internet Engineering Task Force, September 1999.

[rfc2681] G. Almes, S. Kalidindi, and M. Zekauskas. A Round-trip Delay Metric for IPPM. RFC 2681, Internet Engineering Task Force, September 1999.

[diferdinando04] A. Di Ferdinando, P. D. Ezhilchelvan, and I. Mitrani. “Design and evaluation of a QoS-adaptive system for reliable multicasting”. In *Proceedings of the 23rd Symposium on Reliable Distributed Systems (SRDS04)*, Flórianopolis, Brazil, Octiber 2004.

[diferdinandoD8] A. Di Ferdinando, P. D. Ezhilchelvan, I. Mitrani, and G. Morgan. “Qos-aware group communication”. Technical report, TAPAS EU-IST Project, May 2004.

[floyd94] Sally Floyd and Van Jacobson. “The Synchronisation of Periodic Routing Messages”. *IEEE/ACM Transactions on Networking*, February 1994.

[james03] Glyn James. “Advanced Modern Engineering Maths”. Prentice Hall, 2003. ISBN: 0130454257.

[moon99] Sue B. Moon, Paul Skelly, and Don Towsley. “Estimation and Removal of Clock Skew from Network Delay Measurements”. In *Proceedings IEEE INFOCOM*, March 1999.

[rfc2330]V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. “Framework for IP Performance Metrics”. *RFC 2330*, Internet Engineering Task Force, May 1998.

[smotlacha02] Vladimir Smotlacha. “One-way Delay Measurement Using NTP”. In *TERENA Network Conference*, May 2003.

[zhang02] Li Zhang, Zhen Liu, and Cathy Honghui Xia. “Clock Synchronization Algorithms for Network Measurement”. In *Proceedings IEEE INFOCOM*, June 2002.