GloMop Session-layer Preferences and Chunk Scheduling

GloMop and Daedalus Groups

This document describes how Session-layer Preferences from applications to GloMop map to packet scheduling decisions in the Network Stack.

1.0 Overview

1.1 Chunk Schedulers, Network Monitors, and the Network Stack

The chunk scheduler, network monitor, and network stack are the three software components responsible for presenting the abstraction of chunks with priority classes to a client application. There is a chunk scheduler in the GloMop client-side layer and a chunk scheduler in the proxy. The Network Stack also lives in the GloMop client side The Network Monitor may live anywhere in the logical network, but is called primarily from the proxy.

The **Chunk Scheduler** on the GloMop side manages chunk scheduling and multiplexing of chunk transmissions onto multiple physical NI's. The **Network Stack** is responsible for the translation between chunks and network packets. It maintains a transport-level connection between the mobile and the proxy for each available network interface. It handles the fragmentation of chunks into network packets when necessary, and handles reliable transmission of chunks from the mobile to the proxy. It also transmits the chunks according to the priority classes assigned by the Chunk Scheduler. The **Network Monitor** maintains the statistical models that indicate the current performance of each network interface's connection. Note that the actual measurement of the network could be made at the mobile, the base station, the proxy, or some combination of all three. (For most accurate numbers, however, the measurements should be made at the Proxy.) The Network Monitor is responsible for having an accurate summary of the measurement results.

1.2 Network Monitor's Functional Parameters

The actual information kept by the Network Monitor and the messages that the NM supports is specified in the document "Vertical Handoff and Monitoring in Daedalus". See that for details.

I took out link-layer encryption, because the only network that I know of that uses it is CDPD(which no one uses) and if the user really cared, she would encrypt end-to-end anyway.

1.3 Network Stack's Functional Parameters

In addition to any attributes (priority, etc.) associated with a chunk, the Network Stack can have the following parameters turned on and off by the Proxy:

- Whether or not to send chunks reliably. This will change the nature of the transport-layer connction used between the Network Stack on the mobile and Proxy.

1.4 Client-Specifiable Session-layer Preferences

The client application fills in the following template to specify the Session-layer Preferences it would like met. If a template is not specified for a particular chunk, the *subtype-default template* for that chunk type is used instead. Type-default templates are specified at the GloMop level (i.e. global across all client applications) but can be overridden by an application.

The template specifies weighted constraints for the following parameters associated with chunk transcoding and transmission. A weight of zero indicates a parameter whose value is unimportant; a weight of W_{max} means the proxy should go to extreme lengths to meet the bound.

- Maximum end-to-end latency, including distillation and decoding time
- Maximum monetary cost (for metered services)
- Minimum quality (refinement level). As described in the *GloMop Client-Side Architecture*, this can be expressed as vector of refinement-axis parameters or as a scalar on a "gross quality scale". The former only makes sense when the bound is tight, e.g. "Deliver a representation with exactly the following refinement parameters."
- Maximum power consumption. Implementation of this is TBD.
- Encryption: none or end-to-end, applied per chunk. Since encryption is logically orthogonal to other processing we defer discussion of it till later.

For example, specifying an end-to-end latency with a large weight and minimum quality with a small weight has the following semantics: "Make a strong effort to deliver a representation within the specified latency bound, if necessary by sacrificing significant quality."

1.5 Summary of Invariants

- GloMop client deals with nothing smaller than a chunk. The proxy will attempt to create chunks that fit into network packets, but the NS will transparently fragment and reassemble chunks that are too large.
- The NS does not have to provide in-order delivery of chunks. Any ordering is managed by the Chunk Scheduler.
- Reliable delivery can be turned on and off at the chunk level.

2.0 Packet Scheduling

2.1 Priority Classes

The Chunk Scheduler is responsible for assigning a **priority class** to a chunk based on the following deadlines given by the application:

The Network Stack only schedules the transmission of chunks according to priority classes, and does not take part in deadline management.

Lottery scheduling is used to schedule packets belonging to a fixed number of priority classes. Comparisons between priority classes are valid, but absolute performance within a priority class depends on the network characteristics. Sample classes are:

1. Out-of-band urgent: send this notification *now* (e.g. "I have burst into flames")
2. Priority: in-band but high priority (e.g. distilled inlines)
3. First class
4. Second class (like the Post Office)
5. Junk: send when idle (e.g. email)

This or some similar set of default classes will be specified at connection startup time. The Chunk Scheduler may specify new classes by indicating the new class's priority relative to some previously defined class. Although the Chunk Scheduler can specify new priority classes, the number of classes should be relatively static(i.e. don't define a new class for every chunk).

3.0 Using Multiple Physical NI's

When there are multiple NI's, someone must decide which chunks get scheduled on which NI's. There are two options as to who decides scheduling of chunks on NI's, and both have problems:

1. If the Chunk Scheduler decides, there is a violation of layering; the Chunk Scheduler should only present chunks to the network and expect that they get sent to the destination.

2. If the Network Monitor decides, then the statistics that it presents in Section 1.2 must somehow combine characteristics from several (possibly quite different) sources.

We think that the case where the Chunk Scheduler decides which chunks go over which NIs presents less problems, so the Chunk Scheduler will make these decisions.

Given that the Chunk Scheduler makes NI decisions, there still is the question of how to assign chunks to network interfaces. We came up with the following alternatives for the multiple-NI problem:

- Spillover: use some NI (the fastest?) by default; when its expected performance drops below some threshold due to saturation, spill excess packets to the remaining NI's. Measured performance appears to change slowly. This also ignores cost issues.
- By priority class: a particular priority class always uses same NI. Problem: prone to saturation, hard to do fine-grained load balancing. Perhaps a hybrid approach will solve this, which degenerates to spillover when saturation occurs.
- By abstract per-chunk type, which is distinct from the chunk's MIME type/subtype, although the mapping may be static, and is orthogonal to priority class. Type is used to pick an NI, priority is used to order within an NI.

4.0 Other Questions

- What happens with bandwidth reservation schemes when there is not a 1-1 mapping between traffic source and low-bandwidth link: for example, when there are many unrelated proxies competing for the same bandwidth, or (worse) when there is a non-proxy source competing with the proxy for bandwidth?