Use of Erlang/OTP as a Service Creation Tool for IN services

Sean Hinde (One2One)

Abstract

This paper describes an approach taken to Intelligent Network (IN) service development using a combination of standard IN elements (SCP, SSF) and a set of Erlang/OTP nodes using the Mnesia distributed database to store customer records, and the Erlang language to provide complex service logic.

Introduction

The telecoms market has some unique and well documented requirements of its core network systems including:

- Soft real time
- Massively high availability - 99.999%, including planned outages!!
- Enormous complexity - GSM
- Great flexibility

One could probably say without too much fear of contradiction that these requirements are completely orthogonal! The telecoms industry has had to invent some pretty clever tools to allow the flexibility required without compromising the other requirements. Some of these have been hardware based with multiple processors locked together in some level of hot/warm standby, others have been in software engineering techniques, and yet others have been the approach taken by Intelligent Networks.

This approach is to take a high level abstraction of the core building blocks of (in this case) a telephone call. For example a simple freephone call consists of someone picking up the receiver, dialling a number, having this number translated by the network into another number, and routing the call to the destination. IN provides a set of building blocks which allow operators to intervene at each of these points and manipulate the call data at each one in a network safe manner. These building blocks are normally implemented as a gui tool for defining the logic, and normally run on a separate system to the actual telephone exchange.

The availaibility requirements are met by testing the toolset as thoroughly as practicable to ensure that even a misuse of the tools cannot bring down any part of the system, or indeed affect any call other than the one being controlled.

One of the drawbacks of this approach to date has been that the interfaces have never been fully standardised meaning that a vendors IN equipment is much more likely to work seamlessly with its own switches. At One2One we have an entirely Ericsson based GSM switching network and have Ericsson IN Service Control Points.

An architectural decision was taken at a very early stage in our IN depolyment to store the data in an external database and split the logic between the SCP and this database.

The SCP only contains the logic to drive the telecoms interfaces into the switched network and it hands off control of the “Business Logic” to the external database systems which hold the customer data. The SCPs and external databases are interfaced using c7 to TCP/IP protocol converters.

Experience of using a standard UNIX/commercial database.

Initial solutions to this problem of where to store the customer data were (and still are) based on a system written by a small in house team of IT programmers based on C code interfacing to a standard SQL database. Various problems were experienced with this system:

- One read/write failure lead to many others in a row. Solution was to do multi threading!
- Random explosions in the numbers of worker threads. Solution was a full time UNIX admin killing off zombies.
- Commercial Failover system deployed never worked correctly. Took at least 30 minutes to do a manual failover.
- CPU load extremely high. Adding more processors made it worse - turned out to be bug in some spinlock C code.
- Still slowed down periodically - Full time team of DBAs to tune and optimise.

We now have a team of 20 full time C programmers hacking code as fast as they can to try to introduce resiliency into this system.

The Problem

The conclusion drawn from this project by the telecoms designers at One 2 One was that the tools used in the IT software industry are not up to the task of fast and safe development of systems with the properties required in a telecoms network without access to the vast teams of people and months of continuous development and testing available to the telecoms vendors (Surely not - an admission that these things cost a lot to produce?!).

At the time there didn't appear to be many alternatives which were comparable with the existing GUI based IN Service Creation Environments.

The Solution?

Then one day in 1998 while browsing the Ericsson web site for fun the author stumbled across Erlang/OTP. The most noticable quality of this system was that for the first time here was a product which claimed to have almost all the properties we had been looking for in a platform which runs on standard commercial hardware. It proved a little more difficult to get throught the hype and really figure out what the platform was and exactly how it could be used. Permission was obtained to spend some time on an invesigation into whether this platform was the answer to our prayers.

It proved to be very easy to learn to program in a functional style even for a completely non programmer(!), and after a few weeks we had produced the bones of a multithreaded TCP/IP server which would receive a command from the SCP, invoke a thread to implement a piece of custom service logic (e.g. translate a number) and send the result back to the network.

The conclusion to this investigation was that there was minimal customisation required to the standard Erlang/OTP platform to fit in with the existing infrastructure. Namely:

SCP Interface – Socket based using a simple proprietary binary interface
Alarm Interface – to the existing Network Mgt System (socket based text interface)
Stats Interface – ets stats counters needed to be output to files in certain format every 15 mins
Provisioning Interface – Socket based using same protocol as the interface to the SCP
Event Logging – Socket based io of existing event_logger stream to the network management systems
Intranet based monitoring – Simple HTTP query interface to lookup a customer, view alarms, stats.

The effort required to understand the more esoteric parts of the OTP application structure took some more time and effort, but solutions to these customisations were quickly implemented and work started on the first IN service.

Applications

Two large and two small systems have been deployed to date. The first one to go live was the 4th one to be started and the 2nd one to go live was the 3rd one to be started!

Currently in service are:

- **Corporate VPN service**. This service allows short code dialling, global and per VPN black and white lists, Closed User Group functionality (On-net and Off-net calling). It will also present the short code of the calling mobile for a mobile to mobile call. Pretty much all the actual service logic ir written in Erlang rather than on the SCP. There is also a prototype WAP based telephone book service.

  Part of this system is a complete Intranet based customer care system based on the INETS web server included with Erlang/OTP. This has accounted for about 80% of the effort in writing the service. The HTML is currently being ported to a modifed version of the esp Erlang Server Pages system which allows for lists:map/2 loops containing raw HTML and local variables amongst the embedded Erlang code.

- **RADIUS Authentication Server for WAP services.** This system started out life as a database containing a record for all One 2 One customers which would include their IN service profiles. The first requirement turned out to be for authentication of Dial in WAP users so a front end was written in Erlang which implemented the RADIUS protocol. The system consists of 24 erlang nodes – 16 mnesia database servers, 2 O&M, 2 Radius nodes, 2 provisioning nodes, and 2 nodes providing other miscellaeneous functions.

  The database servers are configured as 8 pairs of nodes with data split amongst the nodes using the fragmented tables feature of MNESIA and each fragment replicated across a pair.

  So far the system has been in service for 5 months and has had a series of individual nodes fail for various software licence or hardware reasons but there has not been a single call lost.

Imminently in service:

- **Mini Prepayment system.** This consists of an IN Service which queries an Erlang/OTP based system at the start of the call to allow/disallow the call and at the end of the call to update the customer record with their usage.

  There is an Inets based query front end where one can query the database based on oldest record, or highest usage, or highest number of inbound calls first. It allows an agent to update the cutoff thresholds on an individual basis.

- **Number Translation Service.** This service was the original one planned and due to a whole host of reasons unrelated to Erlang is only now about to go into service.

Issues

In this section I present some of the difficulties and issues encountered along the way (Using Erlang/OTP R6B)

EVA Integration
Integration between the standard EVA application and built in alarm_handler alarms is non existent. In the end I gave up using this and now just send all alarms via SNMP traps using a slightly modified version of the standard alarm_handler. To be useable straight out of the box this would need to be tidied up quite a lot.

Potential for Partitioned Network
The weakest part of the deployed systems is the communication between the database nodes and the potentially serious consequences of a failure for the consistency of the database. A dual LAN arrangement has been set up using some commercial software but there is still a single point of failure in the stacked switched hub arrangement.

It would be nice to see some mechanism whereby multiple sockets could be set up between two nodes using different interfaces with load sharing/failover. Or there could be some retry mechanism before declaring the link to be down.

There must also be scope to perform some form of (perhaps assisted) recovery from a partitioned network situation in the mnesia database.

Memory Usage
The Erlang runtime system can get out of hand in terms of memory allocation. This has manifested itself during testing when a table transform of an mnesia table containing 100k small records exceeded the 1G beam process limit.

Future Work

Future work in One 2 One with Erlang/OTP is currently planned to remain solely in the telecoms domain. Projects currently underway or planned are:

Mass SMS sending tool.

Tuxedo Middleware Interface for query of systems.

Investigation into how the architecture may fit into the future 3rd Generation equivalents of Intelligent Networks

Resourcing

We have found that a single developer can develop and test all of the IN service logic and Erlang code for each of the systems mentioned above within 6 months (some more , some less).

Conclusion

The combination of using an Intelligent Network Service Control Point for driving the telecoms interfaces into a GSM network, and Erlang/OTP based systems to store customer records and provide the complex business logic of the service has proved itself to meet sufficient of the requirements of telecoms network components to be used in full commercial service.

The main reasons this is seen to be true are the existence of the mnesia database system to provide soft real time performance and real time data replication, and the suitability of the Erlang language for writing telecoms logic in such a way that any failures which do occur cannot affect other calls in progress.

Attention to the network resiliency problem and the susceptibility to excessive memory usage would result in a truly formidable system for this sort of database application.