Liberating Asterisk: ADOSA, An Asterisk-inspired Distributed, Open, Softswitch Architecture

J. Penton, A. Terzoli
Computer Science Department
Rhodes University
Grahamstown, 6140
Email: j.penton@ru.ac.za
Tel: (046) 603 8640; Fax: (046) 636 1915
Topic: Next Generation Networks

ABSTRACT - **Distributed, Open, Softswitch Architecture (ADOSA) is a next-generation softswitch solution being designed at Rhodes University. ADOSA is inspired by the Asterisk open source softswitch that has gained significant momentum in the telecommunications industry worldwide. There are many contributions that Asterisk has made to the VoIP and telecommunications industry, one example being its multi-protocol channel architecture. Asterisk, however, has limitations in its design. The idea behind ADOSA is to create a softswitch that inherits many of the core concepts of Asterisk, but within a more scalable, decoupled, distributed approach. This paper introduces Asterisk, discusses its limitations and presents the design of ADOSA.**

I INTRODUCTION

Asterisk is an open source telecommunication software application written in 2000 by Mark Spencer of Digium. Since then it has been growing in functionality and stability on a daily basis with contributions from programmers around the world.

Mark Spencer's intention with Asterisk was to create an application that runs on an Intel-based PC and functions as a Private Branch Exchange (PBX). It was soon realised that Asterisk provided a very flexible telecommunications application in the open source space, which has proved successful in nurturing innovation. Asterisk's multi-protocol architecture, which abstracts telephony applications from telecommunication technologies, and rich feature set have grown its footprint beyond its intended use as a PBX.

Numerous members of the Asterisk community claim Asterisk can be used in the carrier market, replacing class 4 and class 5 switches in the PSTN. Unfortunately, there is still no real evidence to support this claim. This is most likely the result of the telecommunications industry being too conservative to consider an open source platform for carrier environments, but also of some inherent limitations in Asterisk.

ADOSA, Asterisk-inspired, Distributed, Open, Softswitch Architecture, is a softswitch that draws on Asterisk's unique ideas and concepts but extending them to create a distributed, decoupled system that will better suit larger, more complex environments, such as large carriers. Essentially ADOSA uses existing middleware technologies, the Common Object Request Broker Architecture (CORBA), the Simple Object Access Protocol (SOAP) and Web Services, to distribute the tightly coupled Asterisk application.

II ASTERISK

Asterisk's core is a call routing system, connecting calls among users and automated tasks. Calls arrive on various hardware and software interfaces and are routed according to a customisable dial plan. Asterisk provides all traditional PBX features. It features a complete voicemail system and interactive voice response (IVR) system, including music on hold (MOH), as well as basic call services like call transfer, call conferencing, call intrusion, etc. [3], [4].

Asterisk is a standalone application written in C and follows a modular design. Different functions of the server are implemented as modules that are loaded and initialized by a run-time loader.

The main Asterisk application consists of the switching core, scheduler and IO manager, application launcher, codec translator and dynamic module loader. Asterisk's core functionality is accessed easily via four Application Programming Interfaces (API) as illustrated in figure 1.

This work is conducted under the auspices of the Centre of Excellence in Distributed Multimedia at Rhodes University, which is funded by Telkom, Business Connection, Comverse and THRIP.

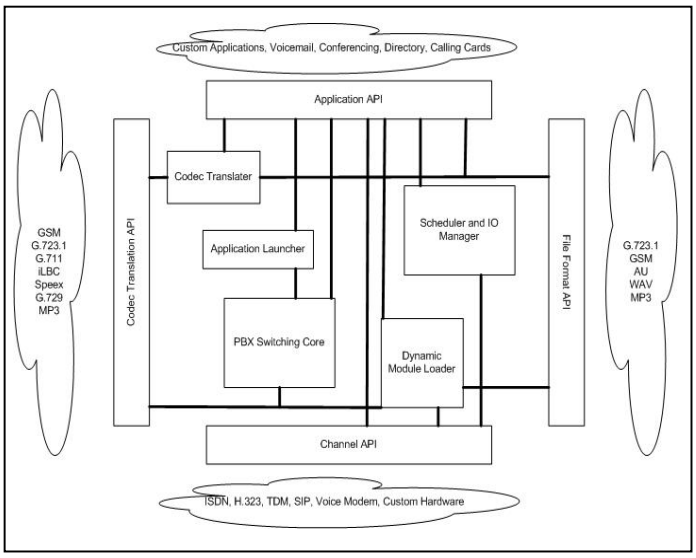

Figure 1: Asterisk application architecture

Channel API:
The Channel API allows the Asterisk switching core to interface with different telecommunication technologies. Via channels, the core functionality of Asterisk is abstracted away from any VoIP, legacy or other protocols or technologies. This means that any application created in Asterisk is accessible by all supported channel technologies transparently. This meta-protocol channel design seems unique to Asterisk and is probably its most interesting design concept.

Codec Translator API:
The Codec Translator API provides a flexible way for the Asterisk core to deal with encoded voice. Formats like GSM, G.723, ADPCM, and MP3 are supported. The codec translator API is responsible for media transcoding between endpoints that do not share a common media format.

File Format API:
The File Format API allows Asterisk to read and play sound in different formats including GSM, WAV, AU, and MP3. This gives Asterisk based applications more flexibility when dealing with ring tones, DTMF and voicemail recordings.

Application API:
The Application API allows developers to interface with Asterisk at any stage of call setup and teardown. The Application API can be used by third party applications, e.g., calling cards, conferencing, and voicemail to take advantage of Asterisk PBX features.

The heart of Asterisk is the Dialplan which specifies the actions the PBX must take based on a number of factors. The switching core follows the directives of the dialplan.

II.1 Asterisk Limitations and Drawbacks

Asterisk is a standalone application that is not designed to be distributed easily among multiple hosts, imposing limitations on scalability, load balancing, failover and management. The Asterisk components and modules illustrated in Figure 1 are confined within the boundaries of the Asterisk application. This, combined with host processing of TDM equipment and software-based DSP functionality, limits the maximum load an Asterisk system can support. The Inter-Asterisk Exchange (IAX) protocol is designed to connect multiple Asterisk servers and can be used to build Asterisk environments that provide rudimentary support for scalability, load balancing and failover. These ad hoc solutions, however, are complicated and inefficient.

Asterisk's tightly coupled design also results in difficulties when managing a multi-node environment. Each Asterisk server in a multi-node system has its own management interface. Each server can be managed individually but managing users, channels and applications across multiple servers is not possible. We have, however, implemented a multi-node Asterisk management system by means of a centralised proxy server [5]. Although this proxy solution works, it is inelegant and ad hoc.

II.2 Asterisk Contributions

Asterisk is a major contribution to the telecommunications industry in its own right. As an open source product, Asterisk provides a cost effective telecommunications product in an extremely expensive market. It also provides young computer scientists and engineers with a feature- and technology-rich environment, in which to learn legacy telecommunication and VoIP protocols and technologies.

Two major technological contributions made by Asterisk to the telecommunications industry are its multi-protocol channel architecture and the Inter-Asterisk Exchange (IAX) Protocol.

The Asterisk channel API, introduced earlier, provides a framework that abstracts the operation of the PBX from a specific telecommunication protocol or technology. This idea seems unique to Asterisk and is the reason Asterisk supports so many VoIP and TDM protocols and technologies.

IAX is a VoIP signalling and media streaming protocol used to signal calls and transport media between Asterisk servers. Naturally, IAX has proprietary support for communicating Asterisk dialplan information, but two attributes of the protocol make it a powerful VoIP signalling and media streaming protocol in its own right. Firstly, IAX combines its signalling and media in the same traffic stream, or port, and does not suffer from the firewall and NAT traversal problems that plague SIP, H.323 and other RTP-dependent protocols. Secondly, IAX outperforms RTP in point-to-point streams by up to 16% and in trunking streams by as much as 45% [10].

IAX is therefore a powerful alternative VoIP signalling protocol and media streaming protocol.

We have included a variant of IAX for signalling and media streaming in ADOSA, which we have called ADOSA's IAX (DIAX). DIAX is IAX without the proprietary Asterisk dialplan functionality and with the inclusion of an appropriate addressing mechanism. IAX is designed to connect Asterisk servers and does not support addressing outside of the Asterisk environment. In our specification of DIAX we have included the Internet-centric method of VoIP addressing currently used by SIP. DIAX is consequently a VoIP protocol that inherits SIP's Internet-centric addressing scheme and IAX's performance and NAT/Firewall traversal attributes.

III ADOSA

Let us consider a basic real-time communication channel. Once established, a real-time call is simply a stream of real-time data flowing between two endpoints. A number of steps and rules are required to setup and tear down a call and these are defined in signalling protocols, e.g. SIP, H.323 and SS7.

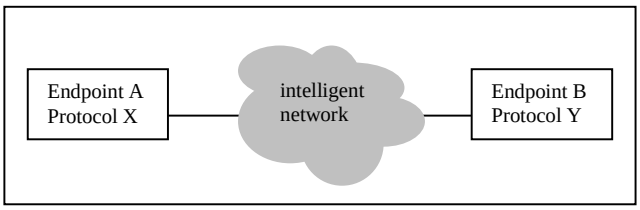

Figure 2: A simple call

Generally, the signalling streams traverse an intelligent network, comprised of entities like proxies, transcoders and gateways. This abstraction is illustrated in figure 2, where endpoint A, using protocol X, is in a call with Endpoint B, using protocol Y. We assume that the difference in signalling between protocols X and Y is handled by the 'intelligent network'. Once the two parties agree to engage in a call the media channels are setup and communication begins.

The primary design goal of ADOSA is to create a framework that encapsulates the functionality of the 'intelligent network', represented by the shaded cloud in Figure 2. The aim is to design a meta-language, or middleware, that facilitates the function and operation of intelligent network entities to provide a robust, scalable and distributed softswitch framework.

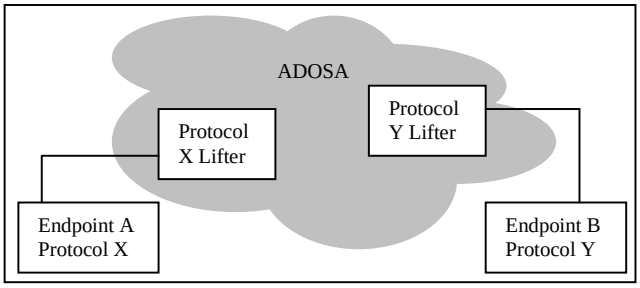

Figure 3: ADOSA lifters

An early assumption in the design of ADOSA is that all endpoints, or edge components, natively communicate with the ADOSA system. To achieve this we introduced specialised modules to translate, or 'lift', existing protocols to a common protocol within ADOSA. In figure 3 two lifter modules are shown that gateway protocols X and Y to ADOSA's signalling protocol, DIAX.

So far we have focused on signalling and have not mentioned the media. Generally, media streams are opened between the communicating parties directly and do not traverse ADOSA, eliminating unnecessary load on ADOSA servers. In some cases, however, the media may have to traverse ADOSA, for example if media transcoding is required. In such a case the lifter is responsible for repackaging the media and re-routing it to the appropriate transcoding module using DIAX.

Naturally, the ADOSA lifter is inspired by Asterisk's multi-protocol channel architecture. The major difference is that the ADOSA lifter is defined as a module located anywhere in the network.

Obvious advantages of ADOSA's distributed lifter architecture are easier scalability, load balancing and failover. Another advantage is that functionality specific to the protocol being lifted can be included in the lifter module. This is in direct contrast with the Asterisk channel architecture, where all channels appear as endpoints. This is the reason Asterisk's SIP channel does not support a fully functional SIP proxy, location and redirect server, for example. Using ADOSA it is possible to add lifter support to the popular SIP proxy server, SIP Express Router (SER), as illustrated in figure 4. The ADOSA-supported SER provides a point of entry for a SIP domain into ADOSA. All calls coming into SER can be offloaded to ADOSA easily, without sacrificing functionality in the SIP network or in ADOSA.

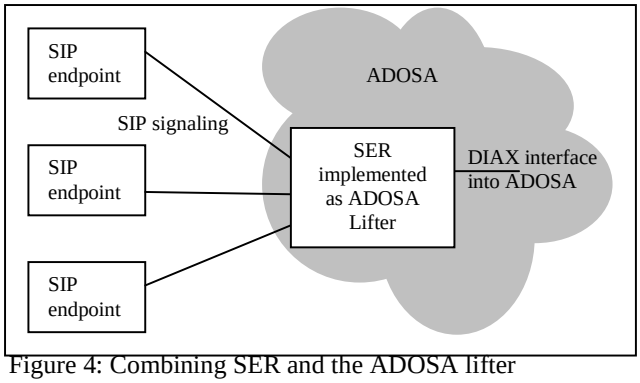

Figure 4: Combining SER and the ADOSA lifter

Furthermore, in Asterisk each call arriving via the SIP channel has its own RTP media stream. The scenario illustrated in figure 4 is highly efficient when there is a large number of simultaneous calls because of the multiplexed trunking design of DIAX.

An implementation benefit of the ADOSA lifter is that it is decoupled from any programming language or operating system. This allows programmers to implement lifters in the

programming language and operating system environment that best suits the solution. Naturally, this is true for any ADOSA module.

IV ADOSA ARCHITECTURE

Figure 5 illustrates the ADOSA architecture, which is divided into two parts, the core component and the peripheral component, represented as shaded and clear blocks respectively.

The ADOSA core is responsible for functionality similar to that provided by Asterisk's core, e.g., call routing, transaction management, scheduling and IO management and application launching.

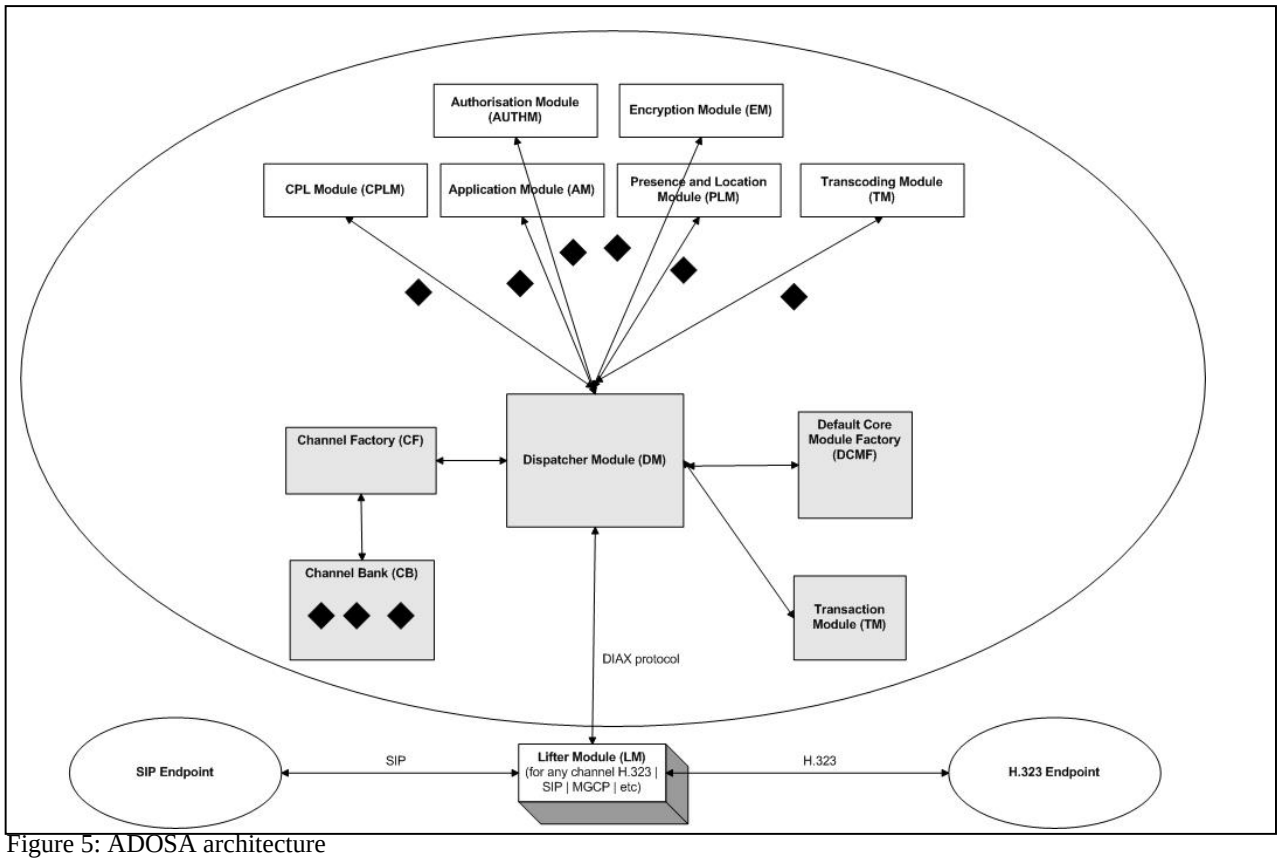

Figure 5: ADOSA architecture

The ADOSA core consists of the following components:

- Dispatcher module (DM) – the central ADOSA switching module responsible for routing communication between system modules.
- Transaction module (TM) – provides scheduling and transaction management.
- Channel Factory (CF) – responsible for the creation and accounting of channel objects. This module is aware of the load the system can handle in a specific configuration and of the number of active channels. Consequently, it plays a major role in the scalability of the system and load balancing within it.
- Channel Bank (CB) - channel banks house channel objects that encapsulate the state and functionality of active channels in the system. Channel banks can be queried and updated by all system modules.
- Default core module (DCM) – the ADOSA core requires functionality that must be available permanently. This module provides such functionality if peripheral modules are not available to satisfy the required functionality.

The peripheral portion provides functionality similar to that provided by the dynamic Asterisk API modules. The peripheral modules facilitate the addition of features, services and functionality to the ADOSA core. Examples are codec modules, calling card application modules, conferencing modules, authentication, authorisation and accounting (AAA) modules. Typically, developers will add to and alter the functionality of ADOSA via peripheral modules.

V ADOSA DOMAINS

Domains are important entities in data networks and it is only fitting that they appear in telecommunication products as they move from legacy circuit networks into domain-centric data networks. ADOSA is designed with the concept of a domain in mind and it is possible to imagine a worldwide ADOSA network made up of smaller interconnected ADOSA domains as illustrated in Figure 6.

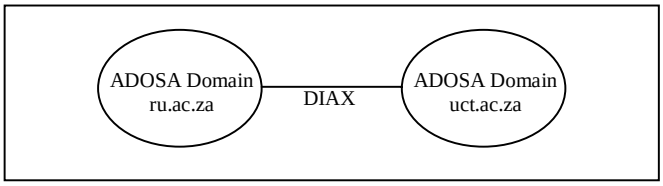

Figure 6: ADOSA domains

For inter-domain routing and control we have specified the use of the DIAX protocol because of its efficient trunking attributes and easy traversal of NATs and firewalls.

VI ADOSA OPERATION

The ADOSA core is the entry point of communication for the lifters. Communication arriving from the lifters is received by the Dispatcher Module (DM) using DIAX. The DM is the heart of ADOSA and dispatches control of a channel to the appropriate modules based on customisable routing logic. For example, a registration request from a SIP client via a SIP lifter is dispatched to a presence and location module (PLM) (peripheral module). The PLM updates the presence and location information for the appropriate user and generates an event back to the DM to send user status messages to the endpoints that have subscribed to the particular user's presence status. Alternatively, if a SIP client makes a call via a SIP lifter, the DM transfers control of the channel to the Call Processing Module (CPM) for routing to another endpoint or application module.

When a new call (or event) enters ADOSA from the lifters, a channel is created. Each channel in ADOSA is represented by an object that contains information about the channel and its status. Channels are created when the DM requests a new channel from the Channel Factory (CF). The CF keeps track of the number of channel objects active in the system as well as the total number of channel objects available. If there is sufficient resources a new channel object is created and stored in one of the available channel banks (CB). Using this architecture it is very easy to scale an ADOSA installation: when the load on the system increases above a specified threshold a new channel bank server can be added to the system and registered with the Channel Factory.

The DM is also responsible for reporting system events to the transaction and scheduling module (TSM) for transaction management. The TSM also serves as a scheduling manager and can trigger interrupts on channel objects when necessary.

The default core module factory (DCM) houses modules essential to the operation of the ADOSA core. If any of the peripheral modules are unavailable, the system can still function as long as the required modules are available in the DCM.

When necessary, the core relinquishes control of a channel object to the peripheral modules. For example, support of a audio or video codec in ADOSA requires transcoding modules. These transcoders can be deployed as peripheral modules that take control of a channel and provide the coding and decoding functionality on their behalf.

The peripheral modules are objects registered with a trading service. When deployed, a peripheral module has a maximum number of simultaneous calls specified, to prevent the server that hosts the module from running out of resources. Multiple modules of the same type can be registered with the trading service. When the DM requires a particular module, it queries the trading service for a module that satisfies the required functionality and has the capacity to serve another channel. As an example, assume a particular ADOSA installation supports the G.711, G.723.1 and G.729 audio codecs. To support these codecs the ADOSA network requires at least three transcoding modules, G.711⇔G.723.1, G.711⇔G.729, G.723.1⇔G.729. Transcoding is a resource intensive operation and any server configuration has a finite number of transcoding instances it can serve simultaneously. If each transcoding server handles a maximum of N simultaneous channels, then when the server starts it registers with the trading service specifying two attributes: the codecs it is capable of transcoding, e.g. G.729⇔G.711, and the number of simultaneous channels it can service, in this case N. Whenever a call requires transcoding the trading service decrements the simultaneous channels attribute by one. In this way no transcoder will be overloaded. Also, if the number of active channels approaches the maximum, the system can generate an alert that will inform the administrator that new transcoding servers may be required. The addition is a simple matter of deploying and registering a new server with the trading service.

VII IMPLEMENTATION PLATFORMS

Given the main design goal of ADOSA, to 'open' Asterisk's tightly coupled design, we had to choose an appropriate middleware for its implementation. We considered the two major distributed object technologies, the Common Object Request Broker Architecture (CORBA) and the more recent Web Service architecture. CORBA is a mature, widely used technology used in many large, robust and mission critical distributed applications. Web Services are relatively new and are not yet as mature as CORBA. Still, we wanted to consider the Web Service technology in ADOSA because it is quickly becoming an Internet industry standard that is reshaping the current human-centric Web to a more application-centric Web [7].

We decided to use both technologies in ADOSA. We chose the Common Object Request Broker Architecture (CORBA) to provide the platform for the ADOSA core component, while a combination of CORBA and Web Service technology is specified for use in the peripheral component of ADOSA, although this decision may be reviewed at a later stage.

VIII CONCLUSION

ADOSA is an attempt at redesigning the popular Asterisk open source PBX. The Asterisk-inspired concepts and ideas behind ADOSA's architecture make it as flexible and functional as Asterisk, but more suitable in larger, more complex environments, like carriers.

IX REFERENCES

[1] *Asterisk,* www.asterisk.org

[2] SIP Express Router, www.iptel.org/ser

[3] J. Penton, A. Terzoli, *Asterisk: A Converged TDM and Packet-based Communications System*, South African Telecommunications Networks and Applications Conference, September 2003, Fancourt, South Africa

[4] J. Penton, A. Terzoli, *iLanga: A VoIP-based, TDM-enabled PBX*, South African Telecommunications Networks and Applications Conference, September 2004, Spier, South Africa

[5] J. Hitchcock, J. Penton, A. Terzoli, *The design of a graphical front-end for an Asterisk-based software PBX*, South African Telecommunications Networks and Applications Conference, September 2004, Spier, South Africa

[6] U. Lang, R. Schreiner, *Developing Secure Distributed Systems with CORBA*, Artech Chouse, Boston, London, 2002

[7] E. Cerami, Web *Services Essentials*, O'Reilly and Associates, U.S.A, 2002

[8] M. Spencer, F.W. Miller, *IAX Protocol Description*, Available online at http://www.cornfed.com/iax.pdf, March 2004

[9] F. Orthman, 2002, *Softswitch Architecture for VoIP*, McGraw-Hill Professional

[10] Convergence Business Systems, *IAX2 Trunking Statistics*, Available online at http://www.convergence.com.pk/iax2/trunked.html, January 2005

Jason B. Penton is a Telkom S.A employee who is currently reading for his Doctoral Degree in Computer Science at Rhodes University. He is working in the area of VoIP focusing on SIP, H.323 and next generation softswitches.