ANGEL Protocol
ANGEL CPE/ISP Protocol Document

Jason But
Centre for Advanced Internet Architectures, Technical Report 070228B
Swinburne University of Technology
Melbourne, Australia
jbut@swin.edu.au

***Abstract*—The Automated Network Games Enhancement Layer (ANGEL) project aims to leverage Machine Learning (ML) techniques to automate the classification and isolation of interactive (e.g. games, voice over IP) and non-interactive (e.g. web) traffic. This information is then used to dynamically reconfigure the network to improve the Quality of Service provided to the current interactive traffic flows and subsequently deliver improved performance to the end users. Within this scope, the project will develop protocols that allow the adjustment of Consumer Premise Equipment (CPE - eg. cable/ADSL) configuration to provide better quality of service to interactive flows detected in real-time.**

This document describes the protocol to be used between ANGEL enabled Consumer Premise Equipment (CPE) and the ANGEL ISP System. The purpose of this document is to ensure inter-operability between ANGEL enabled components developed by different vendors. The ANGEL Prototype system utilises this protocol.

I. Introduction

The primary purpose of the ANGEL system is to detect network game flows within the ISP network and to forward this information to relevant CPE devices which can then take further action such as prioritisation of those flows. The ANGEL system specifically does not enforce how these flows are to be detected, nor how the information should be processed by the CPE devices to allow for integration of new methods and techniques within the ANGEL framework. However, to ensure this continued functionality, and to enable multi-vendor ANGEL systems to be deployed, the protocol describing communications between the CPE and ISP side ANGEL devices must be standardised. This document describes this protocol.

From February 2007 and July 2010 this report was a confidential deliverable to the Smart Internet Technologies CRC. With permission it has now been released to the wider community.

A. Specification of Requirements

The key words **"MUST"**, **"MUST NOT"**, **"REQUIRED"**, **"SHALL"**, **"SHALL NOT"**, **"SHOULD"**, **"SHOULD NOT"**, **"RECOMMENDED"**, **"MAY"** and **"OPTIONAL"** in this document are to be interpreted as described in BCP 14 [1]. These key words mean the same thing whether captilised or not.

An implementation is not compliant if it fails to satisfy one or more of the **must** or **must not** requirements for the protocol it implements. An implementation that satisfies all the **must**, **must not**, **should** and **should not** requirements for its protocol is said to be *"unconditionally compliant"*; one that satisfies all the **must** and **must not** but not all the **should**, and **should not** requirements for its protocol is said to be *"conditionally compliant"*.

B. Terminology

Throughout this document the following terms will be used:

Server - Refers to the ISP Side ANGEL System that is responsible for accepting registration from ANGEL-enabled CPE devices and forwarding information about detected realtime flows to these devices.

Client - Refers to ANGEL-enabled routers, typically CPE devices such as ADSL modems, but also potentially any router in the network that could use ANGEL information

Database - Refers to storage of information about registered ANGEL Clients

II. Operation

The ANGEL Protocol can be divided into two distinct components - the Client Registration Phase and the Server Notification Phase. The Client Registration Phase of the Protocol is entirely responsible for registering and de-registering Client devices and network information within the ANGEL Database. This includes authentication of Client devices as well as managing re-registration time-outs for removal of information from the Database. The Server Notification Phase instead involves communication of information about game flows from the ANGEL Server to the relevant Client Devices. In this terminology *relevant* refers to registered devices through which the specified flow travels - this information is obtained from the Database.

Apart from sharing access to information stored in the Database, the two portions of the protocol can run independently of each other.

In the remainder of this section we outline the procedure through which the parties communicate. Later sections will detail the format of the messages.

A. Client Registration Phase

When a Client is configured to use ANGEL it **MUST** initiate a registration request with the Server. This process involves communicating with the server and **MUST** involve exchanging information about the Client serviced subnets. The provided information will allow the Server to determine which flow notifications to forward to the client.

The Client registration request **MAY** use either the **Open Security Model** or one of the authenticated models. The Server **MUST** reject the request if the requested model is not supported.

Upon receipt of the registration request, the Server:

- If the Client selected the **Open Security** or **Authenticated Model**, the Server **MUST** respond with a registration reply. The Server **MAY** elect to either accept or reject the registration attempt by the Client
- If the Client selected the **Encrypted Security Model**, the Server **MUST** respond with either:
  - A registration reply where the Server **MUST** reject the registration attempt by the Client
  - A Cipher Selection Request where the Server **MUST** maintain a pending based on the response to the **Cipher Selection Pending** packet
- If the **Encrypted Security Model** is used, registration is a two step process where negotiation of a Session Key and associated Cipher **MUST** be performed

ANGEL **MUST** be able to run on systems where clients **MAY** disconnect without running the de-registration phase of the protocol (eg. power disconnection). The Server **MUST** detect the loss of connection and **MUST** remove remove the user registration details from the Database. In order to do this the Server **MUST** inform the Client of a registration *time-out* during which period the Client Registration is deemed to be valid. Once the *timeout* expires, the Server **MUST** remove the Client information from the Database. In order to remain a registered Client, the Client **MUST** re-run the Registration Phase of the Protocol before the *time-out* period expires. In the case of a client re-registration, the server **MUST** reply to the request and either reject the re-registration attempt or accept it with a new registration *time-out* value.

If the Server does not respond to the Client Re-registration request, the Client **MUST** consider that the ANGEL Server is not online and **SHOULD** consider all current flow notifications to be invalid. All current prioritisation rules **SHOULD** be flushed and all traffic treated as best-effort.

The Server **MUST** also respond to the Client with a *flow notification time-out* value. The *flow notification time-out* value is used by the Client to timeout and remove prioritisation rules during periods of flow inactivity. If there is no activity on a (prioritised) flow for the duration of the specified timeout, the Client **MUST** consider the flow to be terminated or re-classified as a typical best-effort flow. This provides stable operation in the event of the Server restarting and ensuring that prioritisation rules do not have an indefinate life.

Also part of the Registration Phase is Client De-Registration. This allows an ANGEL Client to de-register from the ANGEL system and therefore no longer receive any flow notification messages. This phase of the protocol **MUST** be initiated by the Client and involves the sending of a de-registration request. The server **MUST** respond with a de-registration acknowledgement and **MUST** also remove the Client details from the Database.

B. Server Notification Phase

The Server Notification Phase is primarily concerned with the transmission of flow information from the Server to the relevant Clients.

When the Server detects a flow of interest or a change in the classification of a flow, it **MUST** inform relevant Client devices of this information. The Database **MUST** contain information about the flow and its previous classification.

In order to determine which Clients each flow belongs to, the Server **MUST** consult the database and match Flow end-points with registered Client serviced subnets.

In this case the flow classifiincation **MUST** be packaged and sent to the Client(s).

When receiving a flow notification, the Client **MUST** acknowledge receipt. Upon failure to receive this acknowledgement, the Server **SHOULD** resend the flow notification. Upon a change in the classification of a flow, the Server **MUST** send notification of the new classification to the Client, the Client **MUST** use this updated notification and time-out value and properly update its network flow prioritisation scheme. A Server notification **MAY** specify that a particular flow is to be treated as non-realtime - this **MAY** indicate either that the flow is no longer a realtime flow or that it has terminated. In either case, the Client **MUST** remove any prioritisation rules for that flow and force any subsequent matching packets to be treated as best-effort.

The Client **MUST** also (seperately) begin counting down the *flow notification time-out* timer for each prioritised flow. If no packets in the specified flow have been witnessed by the Client during this timeout period then the Client **MUST** assume that the flow has terminated and remove any prioritisation rules for this flow. This technique is used to ensure stability if the Server is restarted and the proper notification of flow termination is not sent to the Client.

The Server **SHOULD** detect flow termination and signal the Client to remove prioritisation rules if a prioritised flow expires. This would typically be done within the Server using a time-out to detect flow termination. This time-out **SHOULD** be shorter than the *flow notification time-out*. This ensures that a flow will be detected to have terminated within the Server *before* the Client. Similarly, if the flow starts up again, it will be detected by the Server causing a new flow notification to be delivered to the Client. A larger time-out may result in the scenario where the Server determines that the flow is still active while the Client considers the flow terminated. In this situation the Client will **NOT** be notified that the flow is realtime.

C. Client Behaviour

An ANGEL-enabled Client is not bound in how it internally responds to a flow notification from the Server. There is no obligation on the Client to prioritise a flow or act on any notifications received beyond acknowledging to the Server that the notification has been received. Eg. some Clients **MAY** have limited prioritisation and/or flow identification capabilities.

An ANGEL-enabled Client **SHOULD** be able to communicate and receive notifications from multiple Servers. In this case the Client **MUST** recognise duplicate notifications generated by multiple Servers and act appropriately when configuring any traffic prioritisation functionality.

D. Transport Layer

The primary purpose of the ANGEL system is to improve the performance of a consumer's (bandwidth limited) Internet connection. While some protocol exchanges are necessary for ANGEL to work, it is of primary importance to minimise the traffic level generated by ANGEL. As such, the ANGEL protocol **SHOULD** utilise a minimal overhead transport layer protocol such as UDP rather than a guaranteed protocol such as TCP.

Since an ANGEL-enabled Client **MUST** have traffic prioritisation capabilities, it is relatively straight forward for the Client to ensure that ANGEL packets (Registration Requests and Notification Acknowledgements) are prioritised for transfer to the Server. This lowers the probability of re-tranmissions by minimising the delay for transfer. Since ANGEL **MUST** use a UDP-like protocol for transport, it is essential that the packet retransmission scheme in the event of packet loss **MUST** be implemented within the higher level ANGEL Protocol.

E. Operational Security Models

The ANGEL Protocol may run in one of three models, more information is given in Section VII. These models are:

Open Security Model: Clients register without username or authentication information. There is no Shared Secret between the Client and the Server. Notification messages received by the Client include a hash to validate the packet contents but do not authenticate the sender of the packet.

Authenticated Security Model: Clients register using a username and secret hash (calculated using a Shared Secret - possibly password based and unique to each Client/customer). The Server responds with an authenticated message calculated using the Shared Secret. Notification messages received by the Client include a hash to validate packet contents, the hash is generated with the use of the Shared Secret to authenticate the Server as the sender of the packet.

Encrypted Security Model: Clients register using a username and secret hash (calculated using a Shared Secret). The Server responds with an authenticated message using a Shared Secret (possibly password based and unique to each Client/customer) and information on the server public key and supported Symmetric Ciphers. The Client must then select a Symmetric Cipher and Session Key for further use. Notification messages received by the Client have their contents encrypted by the Session Key AND are authenticated with the use of the Shared Secret. The Client must decrypt the packet to retrieve notification information. This mode is not recommended due to the computing overhead required by the Client devices.

III. Protocol Reuse

There are numerous existing protocols which may perform some of the functions required by the ANGEL Protocol. In order to minimise development time and to ensure the stability of the Protocol design, we choose to base the ANGEL Protocol on top of the pre-existing RADIUS Protocol [2].

The RADIUS Protocol provides a UDP based packet protocol with the capability of providing authenticated information transfer using a Client/Server model. By re-using the RADIUS design we gain the following major advantages:

- Potential code re-use from pre-existing RADIUS implementations
- Known packet format that can be deconstructed by existing tools
- Existing implementation of Client authentication and authenticated message transfer

The Packet format of the RADIUS Protocol is more clearly described in [2]. This document describes the UDP Port numbers and bit/byte layout of a RADIUS Packet as well as the responses intended to be generated by any requests.

The content and attribute formats for the ANGEL Protocol are described in the following sections.

A. Packet Format

The layout of fields in the ANGEL Packet is identical to that of the RADIUS Protocol.

```
                     1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                         Authenticator                         |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Attributes ...
+-+-+-+-+-+-+-+-+-+-+-+-+-
```

The purpose of each field is:

- **Code** (8 bits) - Identifies the packet instruction. The allowed values and their meanings are described later in this document
- **Identifier** (8 bits) - As per the RADIUS protocol is used to match responses to requests and allows more than one pending request to be made both by the Client (Registration Phase) and the Server (Notification Phase)
- **Length** (16 bits) - The length of the ANGEL Packet stored in network byte order. The length value is the length of all ANGEL Packet fields including **Code**, **Identifier**, **Length**, **Authenticator** and **Attributes**. The minimum value (no attributes) is 20 and the maximum is the maximum possible value that allows the ANGEL packet to fit in one packet (MTU - Network/Transport Header Sizes). Octets within the packet that are longer than the **Length** field **MUST** be treated as padding and ignored. Packets that are shorter than indicated by the **Length** field **MUST** be ignored
- **Authenticator** (128 bits) - Used in a similar way to the RADIUS protocol. The **Authenticator** is a random number during a Client Registration Request, Registration replies must have a properly computed Response **Authenticator**. Also used to store authentication hashes generated by the server when sending a **Notification** packet
- **Attributes** (Length as required - multiple of 8 bits) - Each packet type has a different subset of allowed attributes. Attributes are stored sequentially in the packet structure up to the Length stored in the **Length** field.

B. Attribute Format

Each attribute in an ANGEL packet is also structured into fields. The layout of the fields for each attribute is:

```
                     1 1 1 1 1 1 1 1 1 1 2 2 2 2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|     Type      |    Length     |  Value ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

The purpose of each field is:

- **Type** (8 bits) - Identifies the attribute type. Available attributes are listed later in this document. Attributes **MUST** only be used within the packet type they are valid for. Both Client and Server implementations **MAY** choose to ignore attributes with an unknown type or attributes that are invalid for the packet type. Both Client and Server implementations **MAY** choose to silently discard packets with incorrect or unknown attribute types
- **Length** (8 bits) - The length of the attribute including the **Type**, **Length** and **Value** fields.

- **Value** (Variable - multiple of 8 bits) - The data type and interpretation of the **Value** field is dependent on the attribute **Type**. Information specific to each type is included later in this document

IV. Packet Format - Client Registration Phase

This section highlights the possible packet types that are generated during the Client Registration Phase of the ANGEL Protocol. These packets consist of messages generated by the Client and responses generated by the Server.

As per the RADIUS Protocol [2], transmission of any packets **SHOULD** be repeated if no response is forthcoming from the server. After a configurable number of attempts the Client **MUST** assume that the ANGEL Server is un-reachable and cancel any further attempts. The Client **MAY** choose to retry to connect at a later stage.

Where attributes are listed, the attribute format is as decribed in the definitions in Sections 3 and 6.

A. Client Access Request

This packet forms the ANGEL Client Registration Request. ANGEL provides a number of ways in which a Client can connect to the Server depending upon the required level of security in the overall system. As such, some of the attributes for this packet are optional depending on the required security profile.

Code: 1

Identifier: The Identifier **SHOULD** remain constant across repeated registration attempts but **MUST** be changed on any subsequent re-registration attempts

Length: Length of ANGEL packet inclusive of all fields and attributes

Authenticator: The authenticator is a 16 octet number. As per the RADIUS Protocol, this value **SHOULD** consist of a randomly generated value and **SHOULD** be unpredictable given a previous value for the Authenticator

Attributes: Allowed attributes for a **Client Access Request** packet include:

IP Address

Combined with the **Subnet Mask** attributes, specifies the IPv4 or IPv6 addresses that the Client is interested in receiving ANGEL notifications about. Any flows classified by the server that have either a source or destination IP address in the range specified by the **IP Address** and **Subnet Mask** should have the notification forwarded to this Client. Multiple **IP Address/Subnet Mask** attribute pairs **MAY** be specified to indicate multiple address ranges of interest.

Security Model

Specifies the security model that the Client wishes to use when running the ANGEL protocol. If the server either does not support the requested model **OR** wants Clients to connect using a more secure model, the Client Access Request will fail.

Subnet Mask

Defines the subnet mask to be combined with the **IP Address** attribute to form a range of IP Addresses that the Client is interested in.

Username (Optional)

An ISP may choose to make ANGEL available as an extra (paid for) service to certain customers. In this case there needs to be some authentication to ensure that only authorised users are attempting to register with ANGEL. Authentication details are provided within the **Username** and the Shared Secret provided by the ISP and configured in the ANGEL Client device. This field is required when running ANGEL in either the **Authenticated** or **Encrypted Security Model**.

Secret Hash (Optional)

When running either the **Authenticated** or **Encrypted Security Model**, this attribute must be provided as a means of authentication by the Client to the Server. The Hash is calculated from the **Request Authenticator**, the **Username** and the Shared Secret. The Server will perform the same calculation to verify the identity of the Client.

B. Client Access Accept

This packet forms a successful Server response to the Client Registration Request. If the Client has chosen to use the **Encrypted Security Model** then this packet forms a successful response to the **Cipher Negotiation** packet. This signifies that the Server has received the request, added the Client details to its database and will begin forwarding flow notification messages to the Client.

Code: 2

Identifier: The Identifier **MUST** be equal to the corresponding registration request made by the Client.

Length: Length of ANGEL packet inclusive of all fields and attributes

Authenticator: The Authenticator **MUST** be calculated according to the RADIUS algorithm for determining the Authenticator for a **Access-Accept** packet. The algorithm forms the concatenation of the response **Code**, response **Identifier**, response **Length**, request **Authenticator**, response **Attributes** and the Shared Secret. The MD5 Hash of this string is generated and used to form the Response **Authenticator**.

If running the **Open Security Model** then the Shared Secret is a NULL length string - as such it is possible to forge a valid Response packet.

In either the **Authenticated** or **Encrypted Security Model** the Authenticator will be a unique value that can only be calculated by the Server which **MUST** know the Shared Secret for all possible Clients.

Attributes: Allowed attributes for a **Client Access Accept** packet include:

Session Timeout

A timeout value to indicate to the Client when its registration will be cancelled by the Server. The Client **SHOULD** undertake to re-register (with a **Client Access Request**) within this timeframe. This value forms a heartbeat which is used by both the Client and Server to determine when either system has gone offline.

Flow Notification Timeout

A timeout value to indicate to the Client how long a period of inactivity on a particular flow **SHOULD** be used before considering that flow terminated and removing any prioritisation rules.

C. Client Access Reject

This packet forms a failure response from the Server to the Client Registration Request and can follow either the **Client Access Request** or **Cipher Negotiation** packet. This signifies that the Server has received the request but decided not to register the Client. Possible reasons **MAY** include - but are not limited to:

- Client network range (IP Address and Subnet Mask) invalid for this Client
- Incorrect password if we attempt authenticated access
- Server not accepting new registrations

The ANGEL Client will not get flow notifications from the Server. The Client **MAY** attempt to re-register if the network conditions or configuration information changes. Failure to re-register will result in the Client not receiving any notifications. If registration is rejected the Client **SHOULD NOT** continuously re-attempt when the conditions for rejection have not changed.

Code: 3

Identifier: The Identifier **MUST** be equal to the corresponding registration request made by the Client

Length: Length of ANGEL packet inclusive of all fields and attributes

Authenticator: The Authenticator **MUST** be calculated in exactly the same way as for the **Client Access Accept** ANGEL packet

Attributes: Allowed attributes for a **Client Access Reject** packet include:

Error Code

Signifies the reason for the rejection of the registration request. Can be used to re-configure the Client and/or provide a suitable error message to the user.

D. Cipher Selection Pending

This packet is returned by the Server following a successful completion of the first stage of the registration process when the Client has requested to use the **Encrypted Security Model**. This signifies that the Server has received the registration request, is returning details about available encryption options, and is expecting the Client to choose a Symmetric Cipher and Session Key to be returned in a **Cipher Negotiate** packet.

Code: 4

Identifier: The Identifier **MUST** be equal to the corresponding registration request made by the Client.

Length: Length of ANGEL packet inclusive of all fields and attributes

Authenticator: The Authenticator **MUST** be calculated according to the RADIUS algorithm for determining the Authenticator for a **Access-Accept** packet. The algorithm forms the concatenation of the response **Code**, response **Identifier**, response **Length**, request **Authenticator**, response **Attributes** and the Shared Secret. The MD5 Hash of this string is generated and used to form the Response **Authenticator**.

The Authenticator will be a unique value that can only be calculated by the Server which **MUST** know the Shared Secret for all possible Clients.

Attributes: Allowed attributes for a **Client Access Accept** packet include:

Public Cipher Key

The Server **SHOULD** return the **Public Cipher Key** attribute which defines the RSA

Public Key for the Client to use when formulating the **Cipher Negotiate** packet.

Ciphers

The Server **SHOULD** return a list of all Symmetric Ciphers it supports. The Client can then choose a Cipher supported by both parties

E. Cipher Negotiate

This packet forms the ANGEL Cipher Negotiate Request. This packet is only valid in response to a **Cipher Selection Pending** packet where the **Encrypted Security Model** has been requested. At this stage the Client has a list of Symmetric Ciphers supported by the Server and the Server RSA Public Key. The Client will select a Symmetric Cipher supported by both platforms and randomly generate a Session Key. The attributes of this packet are encrypted using the Server Public Key. In this way the Client selects both the Symmetric Cipher and the temporary Session Key

Code: 5

Identifier: The Identifier **SHOULD** remain constant across repeated Cipher Negotiation attempts but **MUST** be changed on any subsequent attempts

Length: Length of ANGEL packet inclusive of all fields and attributes

Authenticator: The Authenticator should be the MD5 Hash of the string generated by the ANGEL Packet (with the Authenticator set to zero bytes) concatenated with the Shared Secret

Attributes: Allowed attributes for a **Cipher Negotiation** packet include:

Ciphers

Specifies the choice of Symmetric Cipher the Client wishes the Server to use when sending **Flow Notification** packets. This value **MUST** be a cipher that is supported by both the Client and Server. The list of Server supported ciphers are returned in the **Client Access Accept** packet.

While this attribute type supports listing of multiple ciphers, the Server **MUST** accept the first cipher in the list. As such, the Client **SHOULD** only return one cipher in this attribute

Session Key

Specifies the Session Key to use with the chosen Symmetric Cipher. The Client will expect the Server to encrypt all **Flow Notification** attributes with this Key

F. Client De-Registration Request

This packet forms the ANGEL Client De-Registration Request. The Client sends this request to stop receiving notification messages. The same process will be achieved if the Client does not re-register within the **Session Timeout** period except that this allows the Client to de-register before the timeout period expires.

Code: 8

Identifier: The Identifier **MUST** be unique compared to previous Identifiers used in other Client Registration requests

Length: Length of ANGEL packet inclusive of all fields and attributes

Authenticator: As per the **Client Access Request** packet, the Authenticator **SHOULD** be a 16 octet random number. This value **SHOULD** be unpredictable given a previous value for the Authenticator.

Attributes: Allowed attributes for a **Client De-Registration Request** packet include:

IP Address (Optional)

While the Server should be able to determine the Client from the source IP Address of the packet, there is the optional means for the Client to identify itself with the same **IP Address** and **Subnet Mask** used during the registration process

Subnet Mask (Optional)

See above.

Username (Optional)

This attribute is required when running the **Authenticated** or **Encrypted Security Model**. The Client must verify itself to the Server by providing the registered **Username** and re-calculating the **Secret Hash** attribute. The Client **MUST NOT** be de-registered if the verification fails.

Secret Hash (Optional)

See above.

G. Client De-Registration Accept

This packet is returned by the ANGEL Server when the Client requests to de-register. This signifies that the Server has received the request and removed the Client details from its database; no further flow notification messages will be sent to the Client.

Code: 9

Identifier: The Identifier **MUST** be equal to the corresponding de-registration request made by the Client.

Length: Length of ANGEL packet inclusive of all fields and attributes

Authenticator: The Authenticator **MUST** be calculated in exactly the same way as for the **Client Access Accept** ANGEL packet.

If running the **Open Security Model** then the Shared Secret is a NULL length string - as such it is possible to forge a valid Response packet.

In either the **Authenticated** or **Encrypted Security Model** the Authenticator will be a unique value that can only be calculated by the Server which **MUST** know the Shared Secret for all possible Clients.

Attributes: This packet has no attributes.

V. Packet Format - Server Notification Phase

This section highlights the possible packet types that are generated during the Server Notification Phase of the ANGEL Protocol. These packets consist of messages generated by the Server and responses generated by the Client.

As per the RADIUS Protocol [2], transmission of any packets **SHOULD** be repeated if no response is forthcoming from the Client. After a configurable number of attempts the Server **MAY** assume that the Client is unreachable and cancel any further attempts.

Where attributes are listed, the attribute format is as decribed in the definitions in Sections 3 and 6.

A. Rule Notification

This packet forms the ANGEL Rule Notification. These packets allow the ANGEL Server to communicate to the Client which flows would benefit from prioritisation.

Code: 16

Identifier: The Identifier **SHOULD** remain constant across repeated notification attempts but **MUST** be changed on any subsequent rule notifications

Length: Length of ANGEL packet inclusive of all fields and attributes

Authenticator: The Authenticator should be the MD5 Hash of the string generated by the ANGEL Packet (with the Authenticator set to zero bytes) concatenated with the Shared Secret. In the **Open Security Model** the Shared Secret is a NULL String. In the **Encrypted Security Model** the attributes are encrypted by the Server using the Cipher and Session Key negotiated during the Registration Phase.

Attributes: A **Rule Notification** packet specifies a range of attributes to signify a single flow. The order of these attributes is important and all attributes **MUST** be present. If attributes are missing or out-of-order, the Client **MUST** ignore the packet. The Client **SHOULD** also acknowledge receipt of the **Rule Notification** packet so that the Server will not resend the notification.

The correct order of the attributes are:

- Source IP Address - Attribute Type (**IP Address**)
- Destination IP Address - Attribute Type (**IP Address**)
- Source Port - Attribute Type (**Port**)
- Destination Port - Attribute Type (**Port**)
- Protocol - Attribute Type (**Protocol**)
- Category - Attribute Type (**Classification**)

IP Address

The two supplied IP Address attributes are used to indicate the source and destination of the flow being referred to in this message.

Port

The two supplied Port Number attributes are used to indicate the source and destination Transport Layer ports of the flow being referred to in this message. If the flow is an IP flow, then the specified port number **MUST** be 0

Protocol

Specifies which protocol matches the flow being referred to in this message. Valid protocols include IP, TCP and UDP. Other values may be used for future expansion. If the Client operates using an older version of the ANGEL Protocol then the **Protocol** field may contain invalid values - in this case the Client **MAY** either ignore the notification (while still acknowledging receipt to the Server) or it **MAY** treat the notification as an IP notification (**Protocol** = 0)

Category

Supplies the current classification of the specified flow. At present ANGEL only classifies flows into **Game Traffic** and **Non-Game Traffic** categories. If a new notification is received for a flow where the Client has already received a notification, the new notification **SHOULD** be considered current and replace the previous classification for that flow.

B. *Rule Notification Acknowledgement*

This packet forms the response by the Client to the Rule Notification packet sent by the Server.

Code: 17

Identifier: The Identifier **MUST** be equal to the corresponding **Notification** packet sent by the Server

Length: Length of ANGEL packet inclusive of all fields and attributes

Authenticator: The Authenticator should be the MD5 Hash of the string generated by the ANGEL Packet (with the Authenticator set to zero bytes) concatenated with the Shared Secret. In the **Open Security Model** the Shared Secret is a NULL String.

Attributes: The Acknowledgement packet has no attributes.

VI. Attributes

The ANGEL Attributes are used to convey information within the RADIUS encoded packets. For details on formatting of the Attributes for constructing a packet, see the RADIUS Protocol Definition [2]. It is possible for attributes to be listed more than once in a packet, the effect of this is attribute specific.

This section lists all of the ANGEL attributes. Which attributes are required and/or optional in different packet types has already been outlined in the previous two sections.

The Server **SHOULD** ignore attributes of an unknown type. The Server response to a packet with an invalid attribute (for that packet type) **MAY** either ignore that attribute or the entire packet if appropriate.

The Client **SHOULD** ignore attributes of an unknown type. The Client response to a packet with an invalid attribute (for that packet type) **MAY** either ignore that attribute or the entire packet if appropriate.

The value of the **Length** attribute field is inclusive of the **Type** and **Length** fields of the structure and so is always 2 greater than the length of the **Value** field.

A. *Security Model*

This attribute is only valid in **Client Access Request** packets sent by the Client.

Type

0x00

Length

3

Value

8 bit binary value to indicate the ANGEL Security Model the Client wishes to use when communicating with the Server. This version of the ANGEL Protocol supports the following Security Model types:

- **0x00** - Open Model
- **0x01** - Authenticated Model
- **0x02** - Encrypted Model

Other values are reserved for future expansion.

B. *Username*

This attribute is only valid in **Client Access Request** and **Client De-Registration Request** packets sent by the Client.

Type

0x01

Length

Variable (2 + length of value)

Value

An ASCII encoded string containing the username. The string is not NULL terminated and its length is determined from the **Length** field.

C. *Secret Hash*

This attribute is only valid in **Client Access Request** and **Client De-Registration Request** packets sent by the Client.

Type

0x02

Length

18

Value

A 128 bit value in network byte order. Used by the Client to authenticate itself to the server. The value is formed as the MD5 Hash of the concatenation of the **Request Authenticator**, **Username** Attribute and the **Shared Secred**. The Server will perform the same calculation to verify the identity of the Client.
This attribute is only valid in the **Authenticated** and **Encrypted** Security Models where the **Shared Secret** is a non-NULL string.

D. *Public Cipher Key*

This attribute is only valid in **Cipher Selection Pending** packets sent by the Server during the Client Registration Phase.

Type

0x03

Length

Variable (2 + length of value)

Value

Variable length binary value indicating the RSA key used by the Client to encrypt attributes in the **Cipher Negotiate** packet where the Symmetric Cipher and Session Key are chosen.

E. Ciphers

This attribute is only valid in **Cipher Selection Pending** and **Cipher Negotiate** packets.

- **Cipher Selection Pending** - Specifies a list of Symmetric Ciphers supported by the Server
- **Cipher Negotiation** - Specifies the Symmetric Cipher selected by the Server

Type

0x04

Length

Variable (2 + length of value)

Value

A list of byte values where each value specifies a Cipher. This version of the ANGEL Protocol supports the following Ciphers:

- **0x00** - DES
- **0x01** - AES

Other values are reserved for future expansion.

F. Session Key

This attribute is only valid in **Cipher Negotiation** packets sent by the Client during the Client Registration Phase.

Type

0x05

Length

Variable (2 + length of value)

Value

Variable length binary value indicating the Session Key the Server should use with the chosen Symmetric Cipher to encrypt **Flow Notification** attributes

G. IP Address

This attribute is only valid in **Client Access Request** and **Client De-Registration Request** packets sent by the Client during the Client Registration Phase and in **Rule Notification** packets sent by the Server during the Server Notification Phase.

Type

0x10

Length

6 (for an IPv4 Address) or 18 (for an IPv6 address)

Value

A binary field indicating the IP address. The address type is determined from the **Length** value.

H. Subnet Mask

This attribute is only valid in **Client Access Request** and **Client De-Registration Request** packets sent by the Client.

Type

0x11

Length

6 (for an IPv4 Address) or 18 (for an IPv6 address)

Value

A binary field indicating the subnet mask. The address type is determined from the **Length** value. The type must be equal to the immediately preceeding **IP Address** attribute as they are combined to form a range of network addresses.

I. Port

This attribute is only valid in **Rule Notification** packets sent by the Server during the Server Notification Phase.

Type

0x12

Length

4

Value

The port number used to identify a flow. This value is combined with the preceeding **IP address** attribute in a notification packet to signify either the source or destination tuple of a flow

J. Protocol

This attribute is only valid in **Rule Notification** packets sent by the Server during the Server Notification Phase.

Type

0x13

Length

3

Value

8 bit binary value to indicate the Protocol to match a flow tuple against. This version of the ANGEL Protocol supports the following Protocols for flow classification:

- **0x00 - IP** - All IP packets on the specified source/destination IP addresses match this flow
- **0x01 - TCP** - Matches the specified flow tuple for a TCP flow
- **0x02 - UDP** - Matches the specified flow tuple for a UDP flow

Other values are reserved for future expansion.

K. Session Timeout

This attribute is only valid in **Client Access Accept** packets sent by the Server during the Client Registration Phase.

Type

0x20

Length

6

Value

32 bit unsigned binary value stored in network-byte order. The timeout is specified in seconds and signifies the timeout period within which the Client must re-register with the Server following a registration acceptance.

L. Flow Notification Timeout

This attribute is only valid in **Client Access Accept** packets sent by the Server during the Client Registration Phase.

Type

0x21

Length

6

Value

32 bit unsigned binary value stored in network-byte order. The timeout is specified in seconds and specifies the timeout period to use to detect flow termination. If no traffic is seen for a prioritised flow for this period of time, then the Client **SHOULD** consider the flow to have terminated and the prioritisation rule to be invalidated.

M. Category

This attribute is only valid in **Rule Notification** packets sent by the Server during the Server Notification Phase.

Type

0x30

Length

3

Value

8 bit binary value to indicate the classification of the flow referred to by other attributes in the packet. This version of the ANGEL Protocol supports the following categories for flow classification:

- **0x00** - Not classified as a flow that needs prioritisation (default best-effort case)
- **0x10** - Classified as a real-time game flow

Other values are reserved for future expansion.

N. Error Code

This attribute is only valid in **Client Access Reject** packets sent by the Server during the Client Registration Phase.

Type

0xff

Length

3

Value

8 bit binary error code to be returned in event of an error in a request. The code may be used to display or log a human readable error message. Current ANGEL error codes include:

- **0x00** - Requested Security Model not supported by the server
- **0x01** - Required attributes have not been provided
- **0x02** - Authentication error (either the provided Username or Secret Hash is incorrect)
- **0x03** - An invalid IP Address/Subnet Mask pair has been provided
- **0x04** - The requested subnets of interest are not owned by this ANGEL Client
- **0x05** - The selected Symmetric Cipher is not supported
- **0x06** - The selected Session Key length is not valid for the selected cipher
- **0xff** - Unspecified error

A valid ANGEL implementation **MAY** return the *"Unspecified error"* code for any error state INCLUDING other enumerated errors listed above.
Other values are reserved for future expansion.

VII. Security Models

As previously discussed, the ANGEL Protocol can run in one of three different security models. The appropriate model is selected based on the **Security Model** attribute sent by the Client during the registration phase. If the either does not support the model requested by the Client **OR** is configured to not enable that model, then an appropriate Error Code is returned, the Client can then choose to register using a different model or to not use the ANGEL system.

A. Open Model

This model is totally insecure. The Client Registration process is completed without any authentication - the only purpose of the process is for the Server to register which network addresses the Client is managing for purposes of flow notification. In this instance:

- The **Client Access Request** contains only the **IP Address**, **Subnet Mask** and **Security Model** attributes
- The Shared Secret is a NULL string
- Registration responses include the MD5 hash in the Authenticator Field but may be trivially forged (since there is no Shared Secret in input to the MD5 hash)
- Server notification requests include the MD5 Hash in the Authenticator Field but notifications **MAY** be trivially forged

B. Authenticated Model

This model is used to authenticate the sender of ANGEL messages to the receipient - fake packets can be discarded. This model is insecure in that notification messages may be read *off-the-wire* by anyone with appropriate privileges. This is the recommended Security Model with which to operate ANGEL. ANGEL will typically run within an ISP environment with the ISP providing ANGEL services to its customers. In this case the ISP owns the network and it is unlikely that a third-party would have access to ANGEL signalling traffic.

In this model there is a unique shared secret between the Client and Server in the ANGEL Protocol. The registration process includes:

- The **Client Access Request** contains the **IP Address**, **Subnet Mask**, **Security Model**, **Username** and **Secret Hash** attributes
- The **Secret Hash** is calculated using the Shared Secret
- The RADIUS Shared Secret **SHOULD** be unique for each Client
- The Server can authenticate the user using the Secret Hash and Shared Secret
- The Server responds with a **Client Access Accept** with an appropriate response authenticator generated using the Shared Secret
- The Client can authenticate the expected response from the Server using the Shared Secret

Notification packets:

- The authenticator field of the packet is filled with an MD5 Hash of the packet contents and the Shared Secret
- The Client can authenticate the flow notification from the Server using the Shared Secret

C. Encrypted Model

This model is used in a scenario where the contents of ANGEL packets may be accessible to third parties AND where both users and the ISP may not wish to advertise which flows are currently classified as realtime to these third parties. This model is allows both:

- Authentication of the sender of ANGEL messages to the receipient - fake packets can be discarded
- Encryption of attributes in **Flow Notification** packets to each Client so that the realtime flow tuple information cannot be read

This model is not recommended due to the increased processing load required not only by the Server, but more particularly by the Client (CPE Devices) where available processing power is likely to be minimal and packet decryption may not be feasable.

In this model there is also a unique shared secret between the Client and Server in the ANGEL Protocol. The registration process also utilises a Public Key Cipher to negotiate a Symmetric Cipher and Session Key fo encryption of Flow Notification attributes. The registration process proceeds in two steps. The first step includes:

- The **Client Access Request** contains the **IP Address**, **Subnet Mask**, **Security Model**, **Username** and **Secret Hash** attributes
- The **Secret Hash** is calculated using the Shared Secret
- The RADIUS Shared Secret **SHOULD** be unique for each Client

- The Server can authenticate the user using the Secret Hash and Shared Secret
- The Server responds with a **Cipher Selection Pending** with an appropriate response authenticator generated using the Shared Secret
- the **Cipher Selection Pending** includes an RSA Public Key that the Client can use to negotiate Session Keys with the Server and a list of Symmetric Ciphers supported by the Server
- The Client can authenticate the expected response from the Server using the Shared Secret

The second step of registration involves choosing a Symmetric Cipher and Session Key:

- The Client chooses a Symmetric Cipher supported by both parties AND a randomly generated Session Key
- A **Cipher Negotiate** packet is sent to the Server containing the two chosen attributes encrypted using the Server Public Key with the RSA Public Key Cipher
- The authenticator is calculated using the Shared Secret and the MD5 hash
- The Server can authenticate the Client using the authenticator and the Secret Hash
- Only the Server (with its Private Key) can decode the Symmetric Cipher choice and Session Key attributes
- The Server responds with a **Client Access Accept** with an appropriate response authenticator generated using the Shared Secret
- The Client can authenticate the expected response from the Server using the Shared Secret

Notification packets:

- The authenticator field of the packet is filled with an MD5 Hash of the packet contents and the Shared Secret
- Attributes are encrypted using the selected Session Key and Symmetric Cipher
- Only the Client and Server can decrypt the Flow Notification attributes
- The Client can authenticate the flow notification from the Server using the Shared Secret

As the Client registration period expires it is required to re-register with the Server. At this stage the process is repeated and a new Session Key is chosen.

This model may be more appropriately deployed in a an environment where both the contents of the ANGEL signalling protocol may be publically visable AND where there is a need to hide from others which real-time flows are being generated. An example may include a wireless access network with ANGEL deployed on individual Wireless Access Stations.

Acknowledgment

This work was supported from 2005 to early 2007 by the Smart Internet Technology Cooperative Research Centre, http://www.smartinternet.com.au.

References

[1] S. Bradner, “Key words for use in RFCs to Indicate Requirement Levels,” IETF, RFC 2119, Mar. 1997, http://www.ietf.org/rfc/rfc2119.txt.

[2] C. Rigney, S. Willens, A. Rubens, and W. Simpson, “Remote Authentication Dial In User Service (RADIUS),” IETF, RFC 2865, Jun. 2000, http://www.ietf.org/rfc/rfc2865.txt.