Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 5, May 2013, pg.162 – 168

RESEARCH ARTICLE

# Hierarchical EAP Back-End Authenticator State Machine

**Desai Malav[1], Kalpesh Patel[2]**
[1]Department of Information Technology, Gujarat Technological University, Ahmedabad, India
[2]Department of Computer Science and Engineering Gujarat Technological University, Ahmedabad, India

*[1] 13malav@gmail.com; [2] Kalpeshpatel1@yahoo.com*

***Abstract— Authentication is a basic requirement for telecommunication network market for allowing resources to the legitimate users. Extended Authentication Protocol (EAP) is a widely used protocol for authentication. EAP is basically a framework which supports several authentication methods. The several authentication methods supported by a single protocol make this protocol a complex one. To simplify this Complexity State Machines are used. Extended Authentication Protocol (EAP) also has the state machines provided by Internet Engineering Task Force (IETF). 1) EAP peer 2) EAP stand-alone authenticator which is non-pass-through 3) EAP backend authenticator which is used for Authentication, Authorization, and Accounting (AAA) servers, and 4) EAP full authenticator which is for both local and pass-through, are four state machines presented by IETF. These states machines are created using basic single level state machine which reduces the complexity of state machine up to some point. In this paper we will see how we can reduce the architectural complexity using Hierarchical State Machine. Here we will try to represent the hierarchical state machine structure and implement it for the Back-end Authenticator State Machine.***

***Key Terms: - Extended Authentication Protocol; EAP; EAP state machines; EAP peer; Back-end State Machine; Hierarchical Approach***

## I. INTRODUCTION

EAP protocol [5] is typically introduced to provide authentication between different AAA servers [4] or peers with different authentication methods. First, AAA stands for authentication, authorization and accounting [2] [3]. AAA is used in distributed systems for security, allowing access to user and get track about the resources used by the user. This three facility are provided

### *A. Authentication:*

Authentication is the process of identifying the user. This feature provides security by allowing only valid user. This is normally done by USERNAME & PASSWORD. But authentication never related to access right for user.

Biometric Identity is also an example for Authentication.

### *B. Authorization:*

People always confuse the Authentication & Authorization. Authorization is completely different from Authentication. Authentication is “Who the user is?” & Authorization is “What resources he/she can access?” Authorization provides which resources, what facilities can be accessed by user on the basis of user access priorities.

*C. Accounting:*

As the word suggests Accounting is used to make record about consumption by user. Through Accounting we can know about how byte of data user has consumed the amount of time resources are occupied by user. So this data used for trend analysis, cost allocation, billing [6].

Two network protocols providing this functionality are particularly popular: the current working protocol RADIUS [7] [8], and newly introduced Diameter [12] protocol.

*A. RADIUS:*

RADIUS stands for "Remote Authentication Dial In User Service". This protocol is used to provide Authentication, Authorization & Accounting (AAA) facilities. It is client/server protocol runs on the application layer. It uses UDP as transport means RADIUS is sent over UDP.

*B. DIAMETER:*

Diameter is the newer or futuristic protocol which provides AAA functionalities. Here Diameter uses TCP/SCTP as transport. Now in Diameter connection between two Diameter PEERs is established by TCP/SCTP, this is known as transport connection.

In this paper we will first study about Extended Authentication Protocol how the protocol works and supports the different methods. Then we will see EAP peer state machine which is one the state machine defined by IETF. We will see and understand the areas needs to be improved.

## II. **EAP**

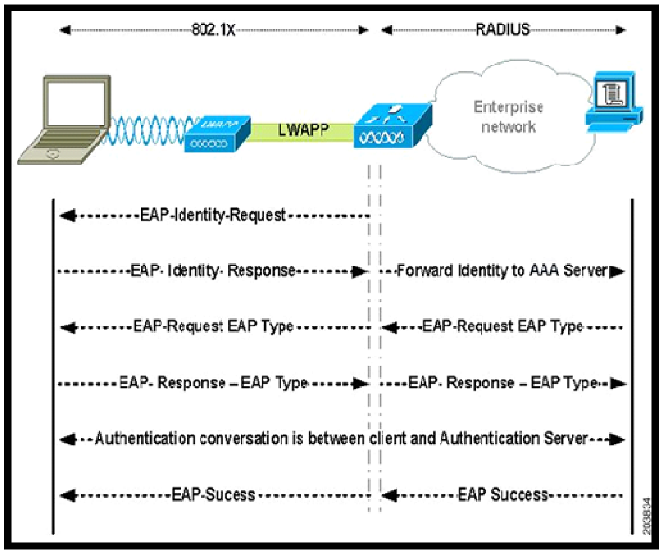


Figure 1 - EAP Packet Flow

EAP stands for Extensible Authentication Protocol. When two peers want to communicate they need a common authentication method to identify each other. But if they both use the different Authentication method than "what to do?" These issue lead to the EAP [9] [10]. Negotiation about authentication method is the most important and most required mechanism provided by EAP. This gives the flexibilities to the server while authenticating. One more and an important benefit of using EAP is if in future a new method is developed for authentication we can use that method through EAP by just adding its plug-ins [10][11]. EAP is an authentication protocol most used in wireless networks and Point-to-Point connections.

The basic EAP is simple and it contains four packet types:

*A. EAP request:*

Each request packet is consisting of type field which gives the knowledge about what is being requested. The authenticator sends the request packet to the peer & fills the type field according to requirement. Request packet also has an 'authenticator' field which helps the authenticator & peer to recognize each other.

### *B. EAP response:*

The response packet is generated to answer the request packet. Generally the response packet contains the same method as it is in the request. If the response packet doesn't contain the same method than it is assumed that response is NAK.

### *C. EAP success:*

After the successful authentication the authenticator sends the success packet to the peer as an acknowledgement of success.

### *D. EAP failure:*

If the authentication failure occurs, the authenticator sends the failure packet to inform failure.

EAP message flow is shown in the figure which shows how authentication is possible between the server based on RADIUS protocol and client based on 802.1x.

## III. **EAP BACK-END AUTHENTICATOR STATE MACHINE DESIGN**

The state machine diagram [11] is shown in the below figure. But to understand the state machine we first have to understand the below specifications used into the state machine.

### A. *Interface between Backend Authenticator State Machine and Lower Layer:*

The data required by the authenticator for processing has been set by Lower layer in the below variable. Also the data required to pass to the Lower layer are also set in this variables by the state machine. There are two categories:

1) *Variables (AAA Interface to Backend Authenticator):*

aaaEapResp (boolean):

Indicates that EAP response packet, stored in aaaEapRespData, is available for further processing by server. If value of aaaEapRespData is NONE indicates server should send the initial EAP request.

aaaEapRespData (EAP Packet):

The EAP packet to be processed, or NONE.

Enabled (boolean):

Indicates that port is available and link is up. If the link is down in any case this flag is set to false.

2) *Variables (Backend Authenticator to AAA Interface):*

aaaEapReq (boolean):

Indicates that a new EAP request is ready to be sent.

aaaEapNoReq (boolean):

The recent EAP response packet has been processed and there is no new request to be sent.

aaaSuccess (boolean):

It is set to TRUE in authenticator when the state machine reaches the SUCCESS state.

aaaFailure (boolean):

It is set to FALSE in authenticator when the state machine reaches the FAILURE state

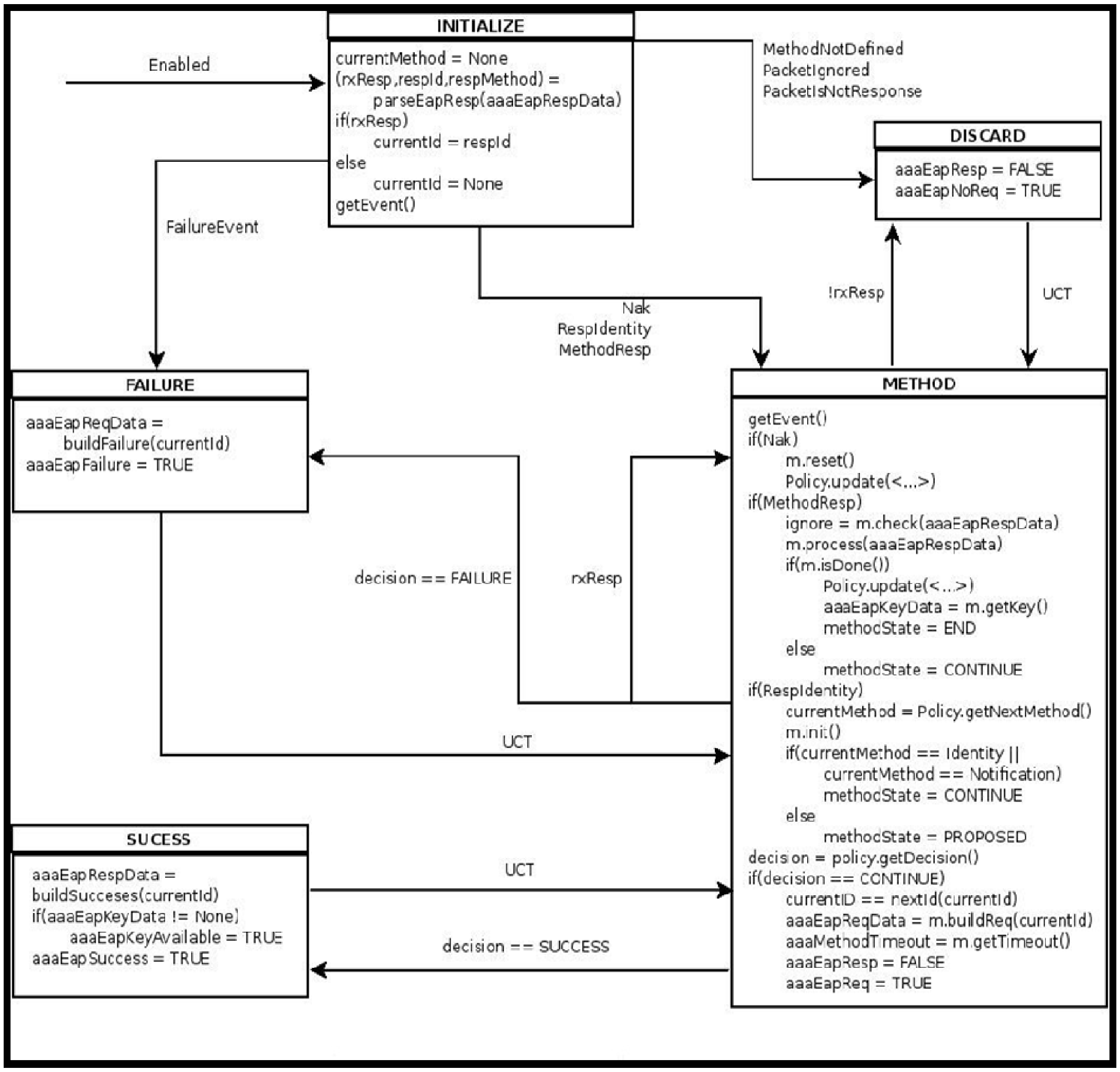


Figure 2 - Modified Back-end Authenticator State Machine

aaaEapReqData (EAP packet):
Set in the authenticator state machine when aaaEapReq, aaaSuccess or aaaFailure set to TRUE. It is the actual EAP request to be sent next.

aaaEapKeyData (EAP key):
It holds the keying material for encryption and decryption.

aaaEapKeyAvailable (boolean):
Set to TRUE in SUCCESS state and it shows that EAP key is available.

aaaMethodTimeout (integer):
It is used to set time on which Retransmission decision is taken.

*B. EAP Backend Authenticator Procedures:*

m.init (in: -, out: -):
It starts the method specific initialization

m.buildReq (in: integer, out: EAP packet):
It creates a new EAP Request packet with the identifier values.

m.getTimeout (in: -, out: integer or NONE):
It can gives the hint for retransmission timeout.

m.check (in: EAP packet, out: boolean):
This method is to check legitimate peer or not. Normally it checks MIC (Message Integrity Check)

m.process (in: EAP packet, out: -):
This method parse and process the request. It returns method state.

m.getKey (in: -, out: EAP key or NONE):
The method processes the EAP Response and updates its own method state. Now it can continue the conversation or it can end it. If authenticator wants to end the conversation

1. Tells the policy about outcome of the method information.
2. If the method found the keying material it will get it from m.getKey().
3. Method calls m.isDone() which return true and method ends.

Otherwise method sends the next EAP Request and continues.

Policy.doPickUp ():
From policy pick up an already-chosen method and process further.

m.initPickUp ():
Initializes the already started method current state and proceed further.

getEvent():
This procedure generates the event which must be occur next and the basis of this event it will go to next state. There are seven events specified:
1. Nak
2. RespIdentity
3. MethodResp
4. MethodNotDefined
5. PacketIgnored
6. PacetIsNotResponse
7. FailureEvent

This all events are based on the actions Extensible Authentication Protocol should take on arrival of the packet. Here Nak includes both Negative Acknowledgment and Negative Expanded Acknowledgment.

*C. EAP Backend Authenticator States:*
There are five basic states is shown below.
INITIALIZE:
At the time of the activation this state is executed and all initialization will occur. The change from the original state machine is here we added the getEvent() procedure. Event generated by this procedure takes state machine to the next state.

METHOD:
This is whole new state. Most of the work is done in this state. The method specific procedures are called from this state machine.

SUCCESS:
A final state indicating success.

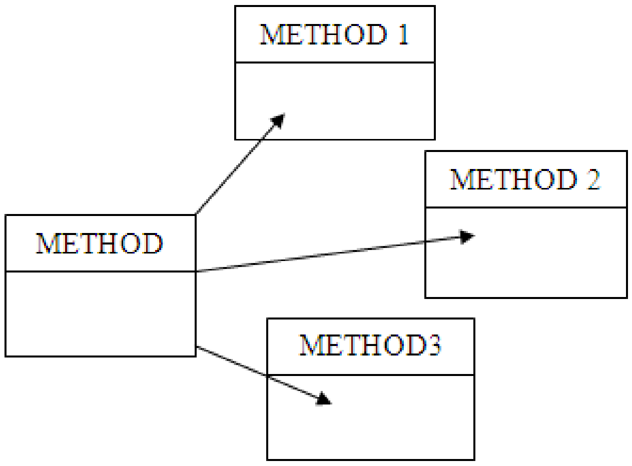
FAILURE:
A final state indicating failure.

DISCARD:
Signals lower layer to discard the packet. This indicates that no new Request is to be sent and further communication is closed.

## IV. HIERARCHICAL STATE: METHOD

Above is all about the modified state machine design. In this state machine all the method specific procedures are in the same state so it will be easy to work on specific authentication method. This state is METHOD state.

In this modified state machine we have deducted multiple flags which cause the change of state. In this state machine this work is done using the Events. These Events are generated by directly using packet data. For example if the eap packet code received is of type Identity (type code 1) than the Event RespIdentity is generated. And handling of the packet is processed accordingly.



The three Events Nak, RespIdentity, MethodResp leads the state machine to the METHOD state. This METHOD state is Hierarchical state. All the method specific state machines, which are authentication methods, are under observation of this state. Call to the method specific state machine and response are handled through this state. The figure gives basic idea about how this works.

## V. CONCLUSION

This new architecture has following qualities over original Back-end state machine.

1. Single state (METHOD) handles the entire method specific state machine so new methods can easily compatible with new methods through ‘PlugIn’. Just add it by implementing a single interface.
2. All the method specific calls are handled through the single state which reduces the implementation complexity.
3. In this state machine State transition is handled using Events so no need to implement extra flags which is able to increase the performance.

## REFERENCES

[1] Lie Han, *A Threat Analysis of The Extensible Authentication Protocol,*School of Computer Science Carleton University, April. 2006.

[2] Christoph Rensing, Martin Karsten & Burkhard Stiller,*AAA: A Survey and a Policy-Based Architecture and Framework*, Darmstadt University of Technology Hasan, University of Waterloo & University of Federal Armed Forces Munich, ETH Zürich, November/December 2002.

[3] Eduardo B. Fernandez and Reghu Warrier, *Remote Authenticator/Authorizer,* Dept. of Computer Science and Eng. Florida Atlantic University Boca Raton, FL, USA.

[4] Christopher Meb,*AAA PROTOCOLS: Authentication, Authorization, and Accounting for the Internet,*Cisco Systems, IEEE Internet Computing November/December 1999.

[5] Jyh-Cheng Chen & Yu-Ping Wang, *Extensible Authentication Protocol (EAP) & IEEE 802.1x: Tutorial and Empirical Experience*,National Tsing Hua University, IEEE Radio Communication December 2005.

[6] Christoph Rensing, Hasan, Martin Karsten, Burkhard Stiller, *A Survey on AAA Mechanisms, Protocols, and Architectures and a Policy-based Approach beyond:* $A^x$, *Computer Engineering and Networks Laboratory,* Swiss Federal Institute of Technology (ETH) Zurich, May 2001.

[7] C. Rigney, S. Willens, A. Rubens, W. Simpson, *Remote Authentication Dial In User Service (RADIUS),* IETF RFC 2865 June 2000.

[8] C. Rigney, *RADIUS Accounting,* IETF RFC 2866 June 2000.

[9] B. Aboba & P. Calhoun, *RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP),* IETF RFC 3579 September 2003.

[10] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson & H. Levkowetz, *Extensible Authentication Protocol (EAP),* IETF RFC 3748 June 2004.

[11] J. Vollbrecht, P. Eronen, N. Petroni & Y. Ohba, *State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator,* IETF RFC 4137 August 2005.

[12] P. Calhoun, J. Loughney, E. Guttman, G. Zorn & J. Arkko, Diameter Base Protocol, IETF RFC 3588 September 2003.