nStack (Neuro Stack):

IEEE 802.15.4

A low rate wireless personal area network specification which defines the physical layer and media access control layer in ISM bands and in certain sub-GHz bands. IEEE 802.15.4 standard offers a low-cost, low-power, low-speed ubiquitous communication between devices. Unlike WiFi, it enables a multi-hop communication mode between devices enabling coverage of longer distances by forming a mesh network.

IEEE802.15.4定义了低速率无线个域网的物理层和媒体访问控制协议，属于IEEE 802.15工作组。运行在2.4GHz或Sub-GHz的ISM频段上，IEEE 802.15.4标准的低功耗、低成本和低速率的特点使其在嵌入式设备通信领域得到广泛的应用。与WIFI不同的是，IEEE 802.15.4能够支持设备间多跳通信模式，通过形成一个Mesh网络极大地扩展通信的覆盖范围。

6LoWPAN (IPv6 over Low power Wireless Personal Area Network):

6LoWPAN enables IPv6 connectivity over low rate WPANs by using stateless compression of IPv6/UDP header. Since the MTUs (maximum transmission units) of IEEE 802.15.4 networks are very limited，thus carrying IPv6 header without compression is impractical. 6LoWPAN adaptation layer is introduced to reduce this header size.

6LoWPAN also defines other primitives such as optimized neighbor discovery operations and fragmentation/reassembly at its layer.

基于IPv6的低速无线个域网

由于IEEE 802.15.4的MTU（最大报文段长度）非常小，未经处理的IPv6协议直接运行在这种协议上非常的不切实际，因为无法携带更多有用的数据信息。6LoWPAN使用无状态IPv6/UDP头部字段压缩，显著减小了IPv6的头部大小（从40bytes到2/3字节），使得IPv6能完美得运行在低速率无线个域网。此外，6LoWPAN还定义了邻居发现机制的优化和分片重组的方法。

RPL (Routing Protocols for LLNs):

RPL (pronounced ripple) is distance vector proactive routing protocol designed especially for low power and lossy networks with interconnections characterized by high loss rates, low data rates and instability of the links. RPL allows creating routing topologies based on different metrics and constraints. nStack optimizes certain aspects of RPL enabling faster network convergence times and faster maintenance operations such as network repairs.

低功耗有损网络路由协议

读作“Ripple”，IETF为低功耗有损网络定义的距离向量路由协议，RPL可以基于不同的路由度量和限制条件创建不同的网络路由拓扑，华为nStack通过RPL路由算法优化，能够使大规模节点组成的网络在短时间内快速收敛，当节点发生故障后，整个网络快速自愈和修复。

CoAP:

CoAP is a RESTful application protocol designed to easily translate HTTP for simplified integration with the web with the constrained nodes and networks.

CoAP是一个轻量级RESTful架构的应用层协议，与HTTP协议类似，但是更加适用于物联网中的资源受限节点。

EAP-PANA (Extensible Authentication Protocol – Protocol for carrying Authentication for Network Access)

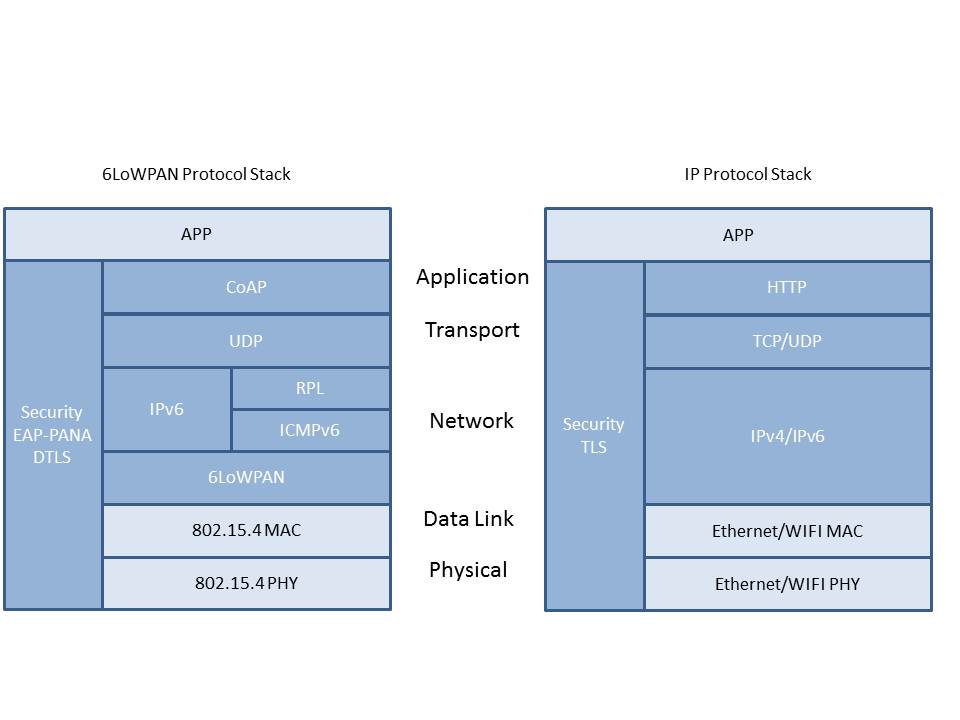
EAP-PANA based security solution enables key management and authentication in multi-hop low power and lossy networks. EAP is an authentication protocol widely used in wireless networks and PANA enables EAP to be carried on IP networks.

EAP-PANA是多跳低功耗有损网络下的密钥管理和认证方案，EAP是一个在无线网络中广泛应用的认证协议（例如WIFI无线链路层使用的认证协议），PANA使得EAP能够运行在IP网络中。

DTLS (Datagram Transport Layer Security)

A transport layer security for UDP based applications. DTLS assumes that the transport connection may be unreliable and the packets may arrive unordered. DTLS is usually used in conjunction with CoAP application protocol for securing the application payload in an end to end manner i.e. between end device and the cloud server.

基于UDP的安全协议，DTLS假设连接可能是不可靠的，或者数据包到达服务器是会产生乱序。DTLS通常与CoAP协议结合使用，加密应用层数据保障传输端到端安全，例如端设备和云端服务器传输数据时。



Socket creation

uint32\_t vpp\_udp\_socket\_register(udp\_socket \*udpSock, void \*ptr, const vpp\_udp\_socket\_input\_callback\_t recv\_callback);

Creates a UDP socket and registers a receive callback. Any incoming UDP message on the socket will be received in this receive callback.

创建一个UDP Socket，并注册一个接收数据的回调函数，任何UDP消息都会在该回调函数中处理。

uint32\_t vpp\_udp\_socket\_bind(udp\_socket \*udpSock, const uint16\_t local\_port);

Bind the created socket to the local port.

绑定Socket到本地的端口。

uint32\_t vpp\_udp\_socket\_connect(udp\_socket \*udpSock, const uip\_ipaddr\_t \*remote\_addr, const uint16\_t remote\_port);

Connect the udp socket to the remote IP address and port. Only UDP messages from the given remote address will be accepted on the UDP socket.

连接UDP的Socket到指定的远端服务器IP地址和端口。注意，连接成功之后，才能接收到指定服务器发往该Socket的UDP报文，并且只能接收到该接口指定远端服务器的UDP报文。

int vpp\_udp\_socket\_send (udp\_socket \*udpSock, const void \*data, uint16\_t datalen);

Send on the UDP socket. For this function to work the udp socket must be connected using a previous call to vpp\_udp\_socket\_connect();

在该UDP Socket上发送数据。发送数据之前，必须先调用vpp\_udp\_socket\_connect()建立连接。

int vpp\_udp\_socket\_sendto (udp\_socket \*udpSock, const void \*data, uint16\_t datalen, const uip\_ipaddr\_t \*remotrAddr, uint16\_t remotePort);

Send on the UDP socket to the given remote IP address and remote port. The udp socket needed not be explicitly connected before using vpp\_udp\_socket\_connect(). Note that send() and sendto() can be used on the same connected UDP socket.

在该UDP Socket上发送数据到指定的远端IP地址和端口。发送数据之前，不需要显示调用vpp\_udp\_socket\_connect()来建立连接。注意，send()和sendto()可以在同一个UDP Socket上使用。

typedef void (\* vpp\_udp\_socket\_input\_callback\_t) (struct \_udp\_socket \*c, void \*ptr, const uip\_ipaddr\_t \*source\_addr, uint16\_t source\_port, const uip\_ipaddr\_t \*dest\_addr, uint16\_t dest\_port, const uint8\_t \*data, uint16\_t datalen);

Callback is to use to register input udp socket

用来接收数据的回调，回调函数在创建Socket时使用。

uint32\_t vpp\_udp\_socket\_close

(

udp\_socket \*udpSock

);

Close the udp socket.

关闭UDP Socket

|  |  |  |  |
| --- | --- | --- | --- |
| Function | nStack | BSD | Remark |
| Socket Create | vpp\_udp\_socket\_register   1. 接收函数作为参数传递给Socket创建函数 2. 仅支持基于IP的接口，类似BSD接口参数中的AF\_INET或者AF\_INET6 | socket | 用户需要创建udp\_socket结构作为参数 |
| Bind | vpp\_udp\_socket\_bind   1. 不需要sockaddr参数，一个端口绑定所有的IP地址 | bind | 需要绑定的端口作为参数传入 |
| Connect | vpp\_udp\_socket\_connect  1，连接成功之后，才能接收到指定服务器发往该Socket的UDP报文 | connect |  |
| Send/Sendto | vpp\_udp\_socket\_send/ vpp\_udp\_socket\_sendto  1，仅支持阻塞模式（Blocking-Mode） |  |  |
| Receive | vpp\_udp\_socket\_input\_callback\_t  1，通过创建Socket是注册为回调函数 | recv  recvfrom | 回调函数 |
| Close | vpp\_udp\_socket\_close | close |  |