6WINDGate Fast Path Plugins

6WIND

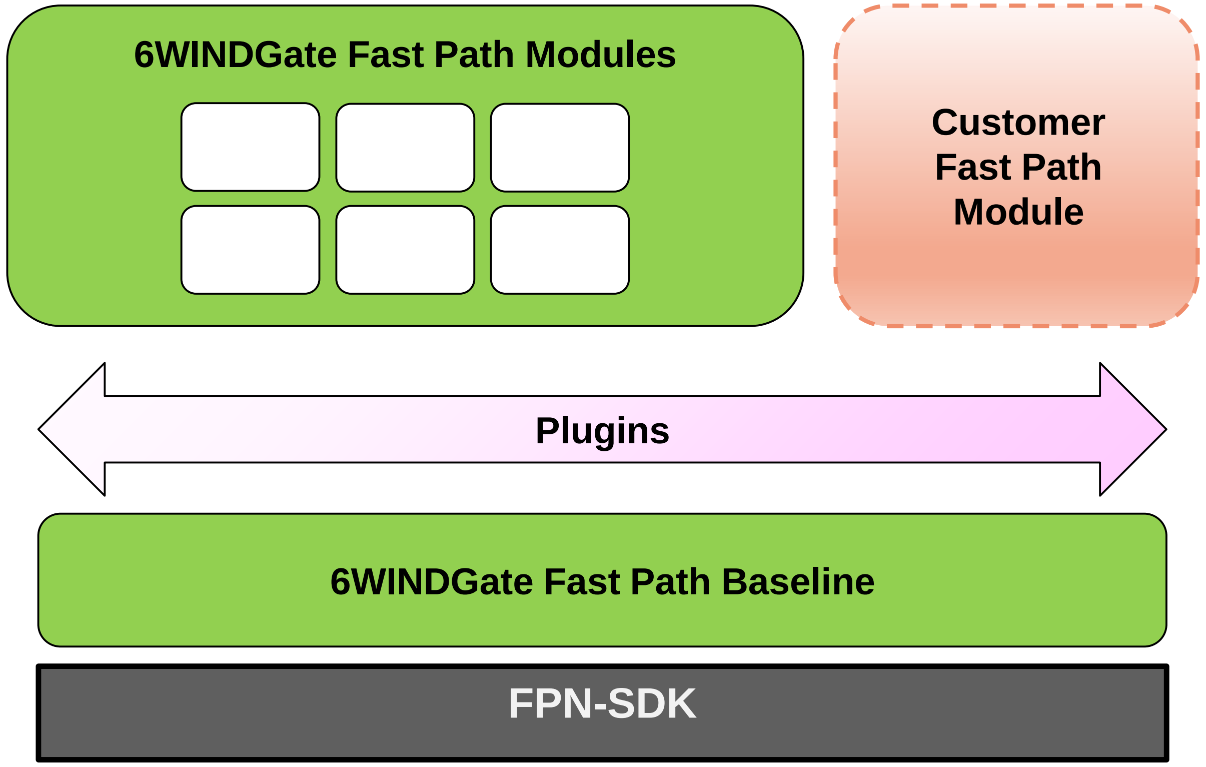
#SPEEDMATTERS For Serious Networks

What Is A Fast Path Plugin?

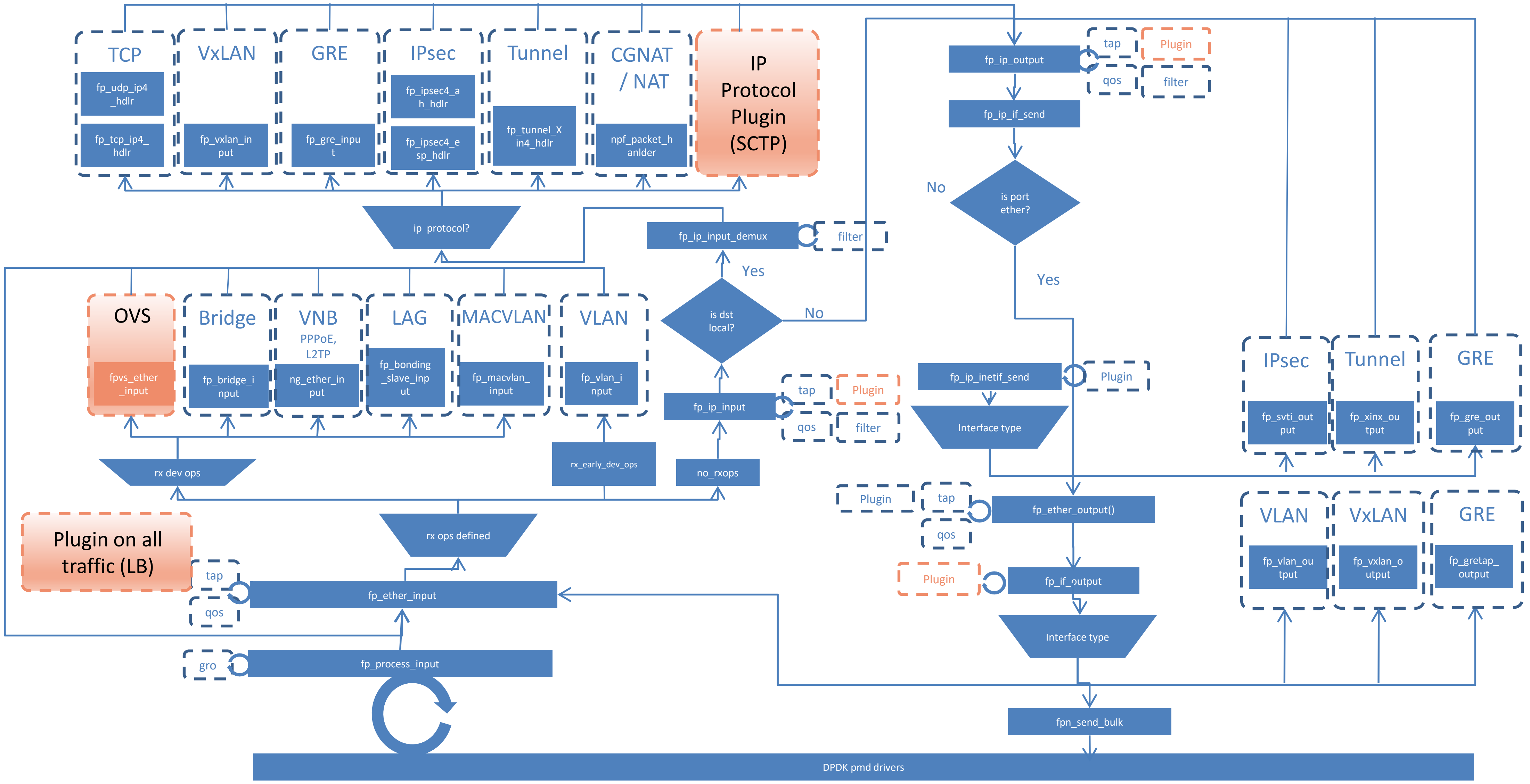
- **Fast Path plugins make it possible to customize some part of the Fast Path application without modifying the original Fast Path modules**

- **Three different kinds of plugins**
  - Replace a Fast Path module with custom processing
  - Replace input and output functions on specific interfaces
  - Handle a new protocol on top of IP

- **Plugins are a feature for the Fast Path Baseline module**


6WINDGate Fast Path Modules
Customer Fast Path Module
Plugins
6WINDGate Fast Path Baseline
FPN-SDK

Three Examples of Fast Path Plugins

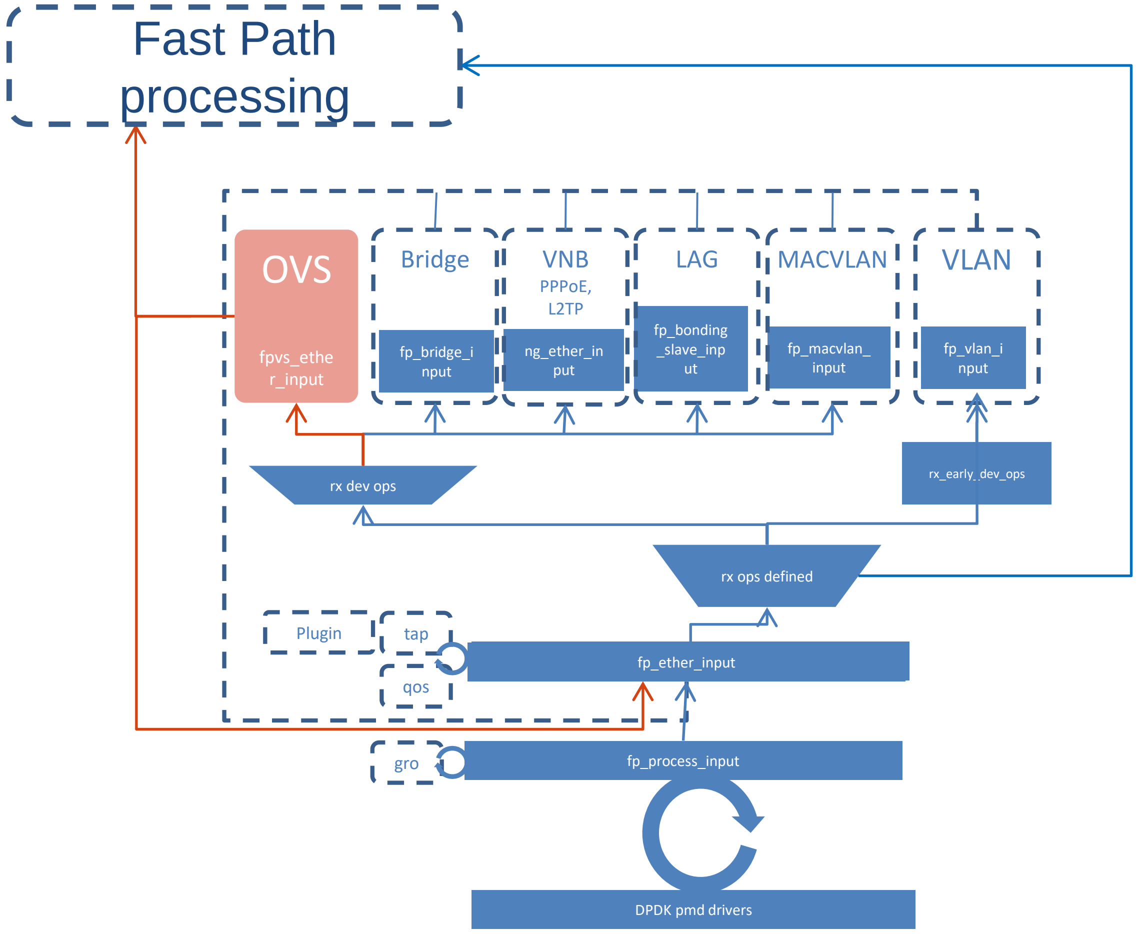


Replace Input and Output Functions on Specific Interfaces

- **A plugin registers its own functions in place of Rx/Tx hooks for a specific interface**
- **Rx/Tx hooks are provided after Ethernet processing in ingress and before Ethernet processing in egress**
- **Interface processing hooks**
  - rx_dev_ops
  - tx_dev_ops

6WINDGate OVS

- **6WINDGate plugin**
- **OVS module defines the RX and TX functions for interfaces that are part of OVS bridges**
  - All packets will be processed by these new functions
  - For example eth0 rx_dev_ops = fpvs_ether_input
- **According to OVS rules the packets will be reinjected to Fast Path or emitted**



Replace a Fast Path Module With Custom Processing

- **A plugin overrides one of the predefined Fast Path hooks to receive packets instead of the corresponding Fast Path module**
- **After plugin processing, the packets can be re-injected in the Fast Path stack using the Fast Path modules input functions**

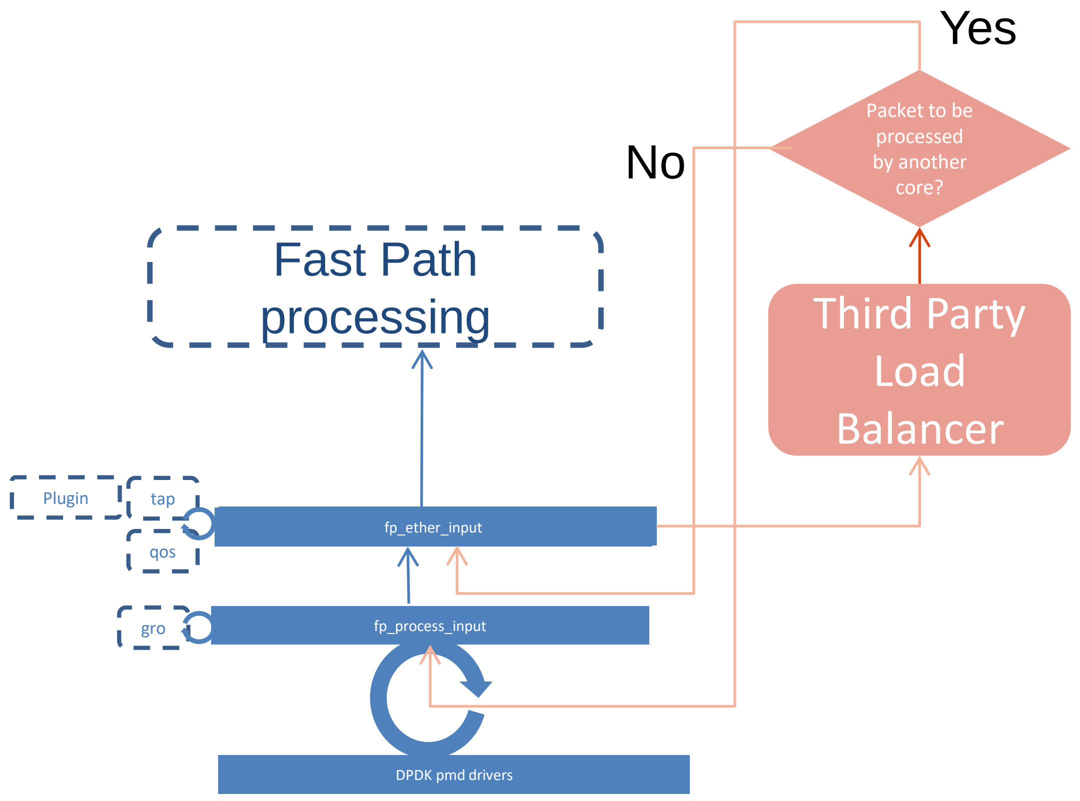
- **Predefined hooks**
  - fp_ether_input
  - fp_ether_output
  - fp_if_output
  - fp_ip_input
  - fp_ip_inetif_send
  - fp_ip6_input
  - fp_ip6_inet6if_send
  - fp_process_linux_tx

Third Party Load Balancer

- **Custom load balancer to dispatch packets for protocols that are not supported by RSS**

- **Load balancer code hooks at fp_ether_input() level**
  - All packets will be diverted to load balancer code

- **Two actions according to load balancer implementation**
  - Packet will be processed by another core
    - It will be sent through inter-core ring and received in main loop through fp_process_input()
  - Packet will be processed by the current core
    - It will be reinjected in Fast Path by calling original fp_ether_input()
  - In both cases the packet processing will be resumed into the original Fast Path code
    - No modification of the Fast Path code is required



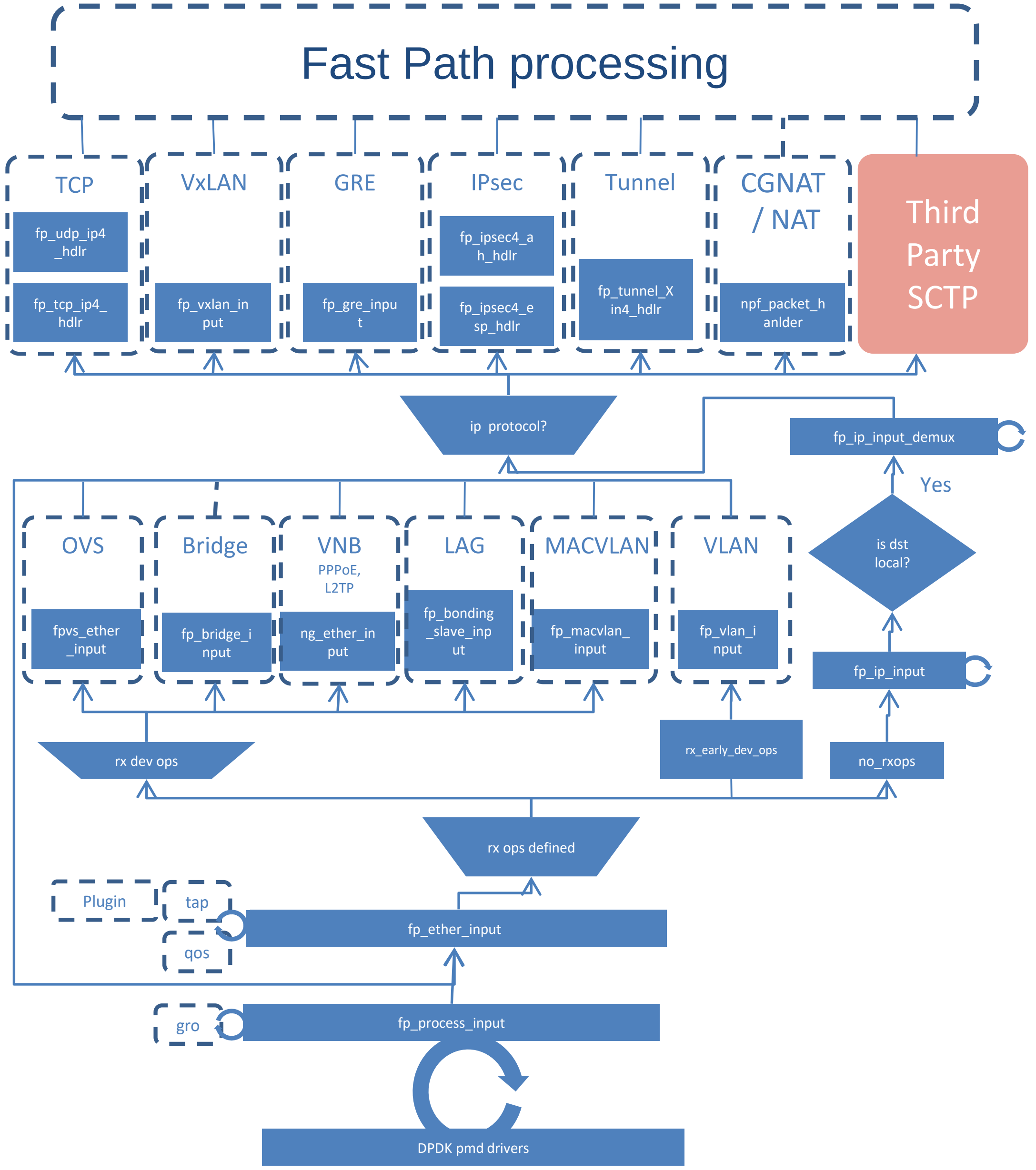
Handle a New Protocol On Top Of IP

- **A plugin registers its own function to handle an IP protocol**

- **IP protocol register function**
  - fp_ip_proto_handler_register(IP_PROTO_ID, handler_function)
  - fp_ip6_proto_handler_register(IP6_PROTO_ID, handler_function)

Third-Party SCTP

- SCTP module registers its handler function for SCTP IP protocol
- When SCTP packets are received with locally configured destination IP address they are processed through this SCTP packet handler
- SCTP packet handler will be responsible to terminate SCTP protocol in Fast Path



Fast Path Plugin Management

- **A plugin can extend the Fast Path CLI by registering its own commands or extending existing fp-cli commands through predefined hooks**

- **fp-cli new command register function**
  - fpdebug_add_commands()

- **fp-cli extension hooks**
  - fpdebug_add_stats()
  - fpdebug_add_ifnet_info()
  - fpdebug_add_runtime_arg()

Fast Path Plugin Synchronization

- **Shared memory to share packet processing information between Linux and the Fast Path**
  - Shared mem API provided by FPN-SDK
  - User to implement Linux-to-Shared-Mem synchronization daemon

- **NETFPC to communicate between the userland and the Fast Path plugin**
  - A plugin can be notified on reception of a NETFPC message by the Fast Path by registering its own handler function
    - For example interface creation, interface flags setting…
    - New NETFPC messages can be defined if required
  - NETFPC message register function
    - fp_netfpc_register(NETFPC_MSG_TYPE, handler_function)

Thank You

6WIND.com

#SPEEDMATTERS For Serious Networks