A Scalable High-Performance Active Network Node

D. Decasper, B. Plattner – ETH Zurich

G. Parulkar, S. Choi, J. DeHart, T. Wolf – Washington U

Presented by Jacky Chu

Motivation

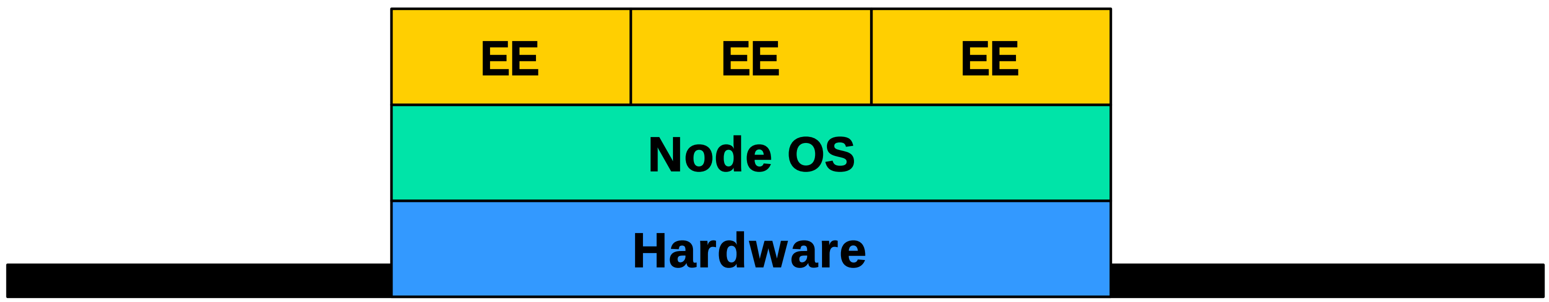
- Apply Active Network over gigabits links is a big challenge.
- Memory bandwidth and processing power of a microprocessor is the bottleneck for an active node in Active Network.
- No microprocessors can singled-handedly solve the problem, and thus an active node requires to be scalable on both hardware and software.

Related Work

- MIT
  - Capsules – Small fragments of code
  - ANTS toolkit – Packets carry pointers
- BBN
  - Smart Packets – Extend diagnostic functionality in the network
- Georgia Tech
  - Shows self organizing network caching extends network's capability
- U Penn
  - Switchware (switchlet) and uses field programmable gate arrays
- U Arizona
  - Scout - fast but not fast suit for high-volume high-bandwidth traffic
- Columbia U
  - Netscript – middleware for intermediate network nodes

Proposing a New Architecture

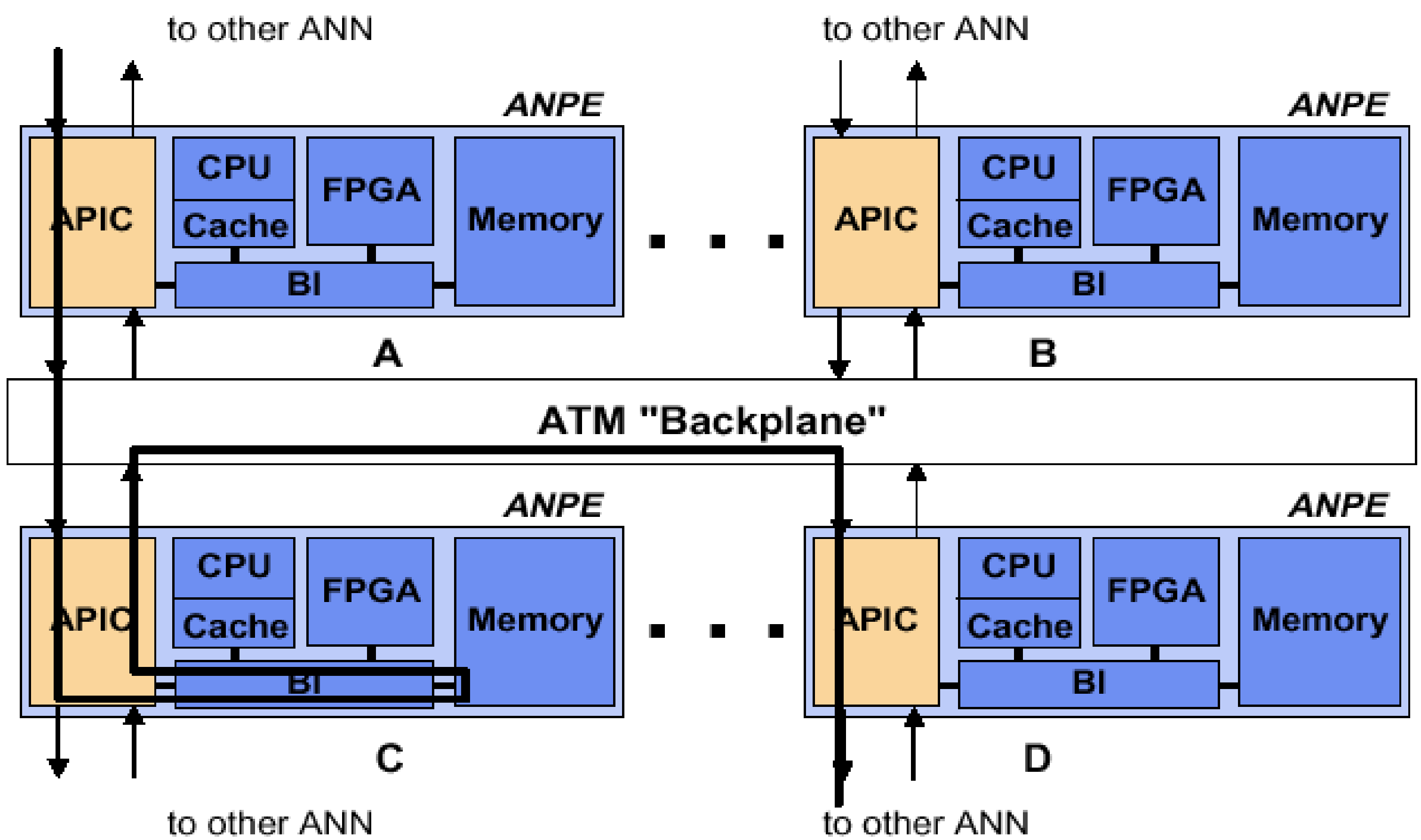
- 3 key components of the architecture
  - Hardware of an Active Network Node (ANN)
  - NodeOS of ANN
  - Execution Environment of ANN



Hardware Requirement

- High number of processing elements per router ports
  - General purpose CPU and a Field Programmable Gate Arrays on every port on backplane
  - CPU does majority active functions, while FPGA takes care performance-critical functions
- Tight coupling between engine and network
  - Network traffic is flow-oriented and burst of packets share common forwarding properties
  - Majority of non-active packets allow cut-through with CPU intervention
- Scalability
  - Evenly distribute computation over processing unit (CPU + FPGA)

Hardware for ANN Overview



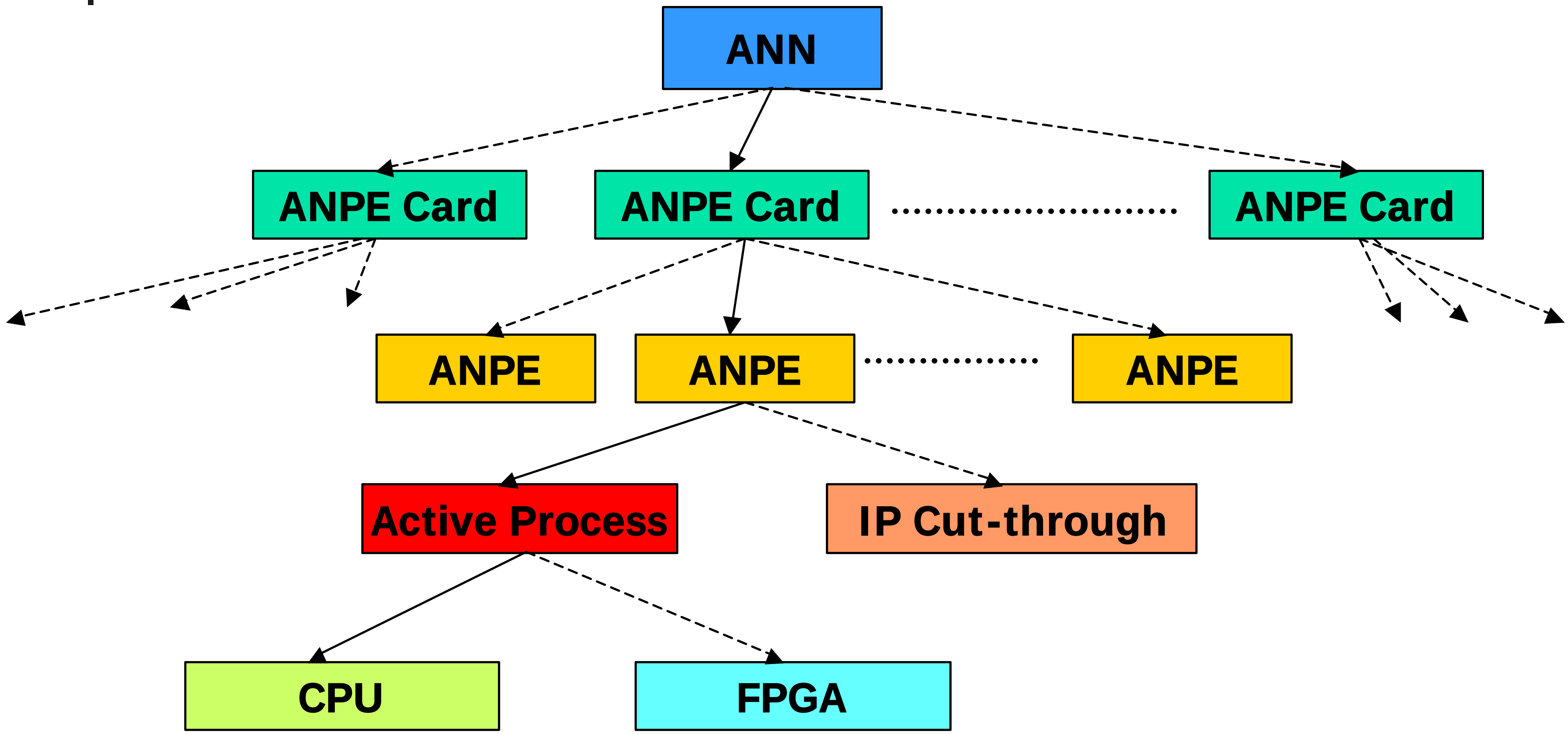
ANPE – Active Node Processing Elements

- ANNs are interconnected through ANPE
- ANPE are connected to the ATM backplane via ATM port interconnect controller (APIC)
- Number of ANPE per ANN is configurable
- Backplane is scalable
- Load-sharing algorithm distribute load by configuring APIC in each ANPE
- Data flow can be routed between ANPE, sharing or pipelining the process on packets

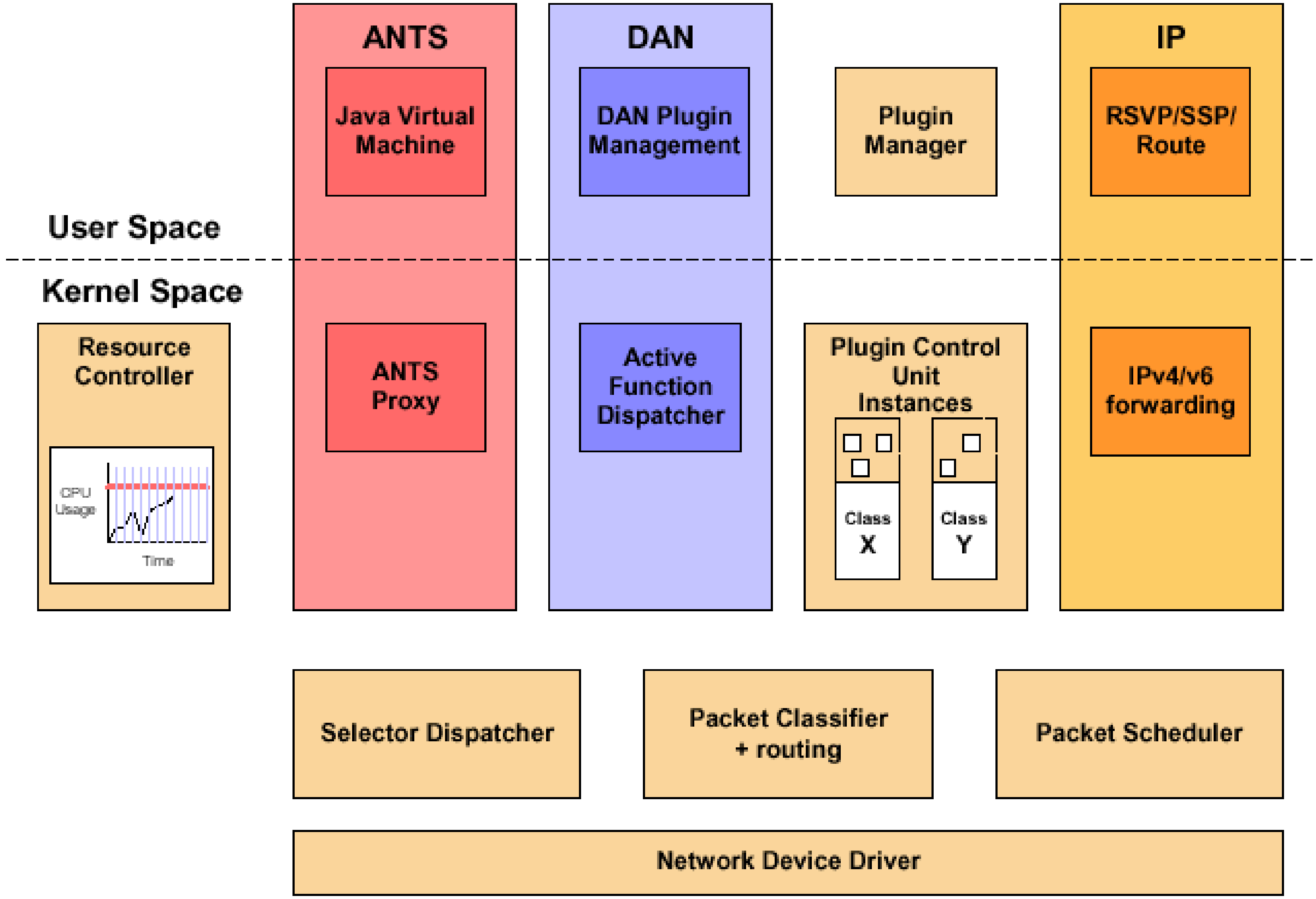
ANPE Details

- APIC – ATM host-network interface, two ATM ports and a built-in PCI bus interface, implementing VC switching (allow load-sharing and cut-through IP traffics).
- CPU runs NodeOS (optimized NetBSD)
- FPGA can be programmed by CPU on the fly
- Packet can go into either CPU or FPGA

Redundancy Provides Scalability



ANN Software Infrastructure



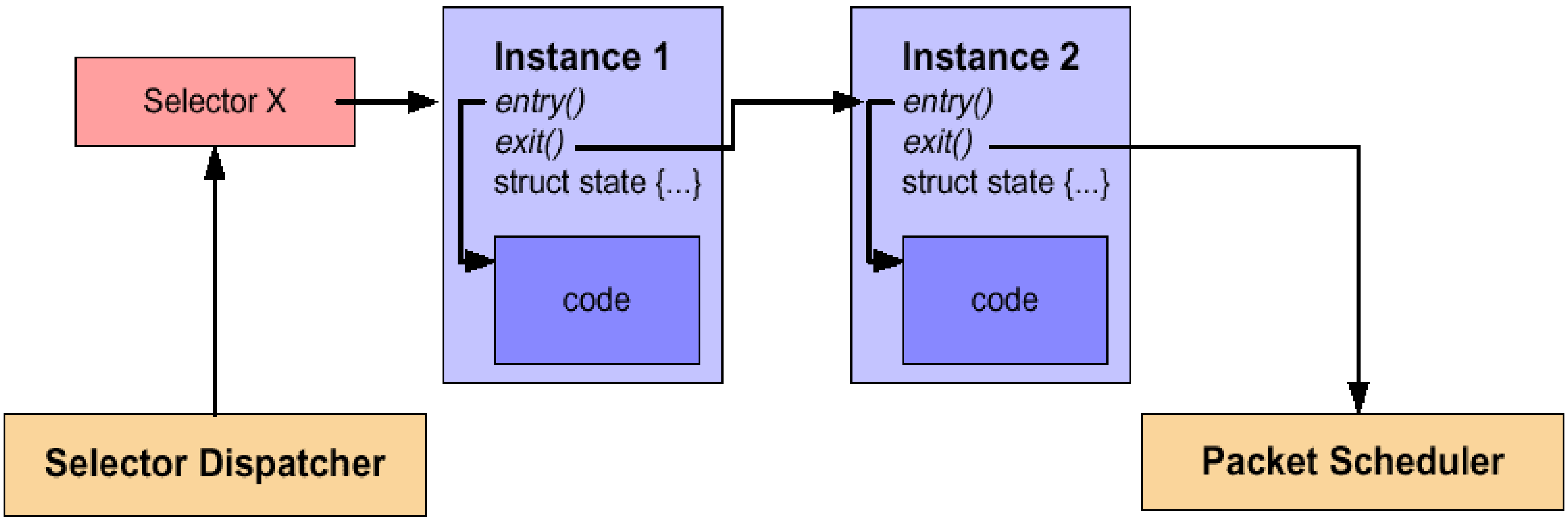
Node OS

- Execution Environment (EE) runs on top on NodeOS, example:
  - ANTS, Smart Packets, SwitchWare
  - DAN – Distributed Code Caching for Active Network
- Code blocks implementing application-specific network functions are Active Plugins.
- Active Plugins can create object instances.
- API **`enter()`** is called to pass packet to instance.
- API **`exit()`** is called by instance when finished processing packet.
- Designed to favor DAN architecture while compatible to other EEs

Object Daisy Chain

- 1st packet of a flow will cause plugin to create instances while the subsequence packets won't.
- If a packet is needed to process by multiple instances, in each instance's **`exit()`** will call method **`enter()`** of the next instance.
- Chains are labeled by a Selector
- Last element of a chain is a Packet Scheduler that send packet back to the network.

Object Daisy Chain



Node OS Components

- Device Drivers (DD)
  - Standard NetBSD on send and receive packets.
  - exception on using packet scheduling instead of using packet queue.
  - If packet contains a selector, it goes to Selector Dispatcher instead of the IP stack.

Node OS Components

- Packet Classifier (PC)
  - For packets without selector
  - Every new flow has a flow record
  - Classified by src and des {IP, port} and protocol
  - Tags all incoming packets with flow index (FIX)
    - contains MBUF and points to flow record
  - Plugins can access the flow record by FIX

Node OS Components

- Selector Dispatcher (SD)
  - Scans packet for selector
  - Use selector to find FIX
  - Stores a pointer to the first instance of the chain
- Packet Scheduler (PS)
  - Can use both Deficit Round Robin or Hierarchical Fair Service Curves for scheduling packets

Node OS Components

- Resource Controller (RC)
  - Responsible for fair CPU time sharing
  - Implements selection of a queue
  - RC in an ANN exchanges information with each other over a reserved VC to provide input for the load-sharing algorithm
  - Also keep tracks of memory consumption on per-instance basis
  - Can deny additional memory demanded by greedy instance

Node OS Components

- Plugin Control Unit (PCU)
  - Manages plugins
  - Forwards control path messages,
    - E.g. instance creation, registration information
- Plugin Manager (PM)
  - User-space utility to configure system
  - Provide interface to kernel space components

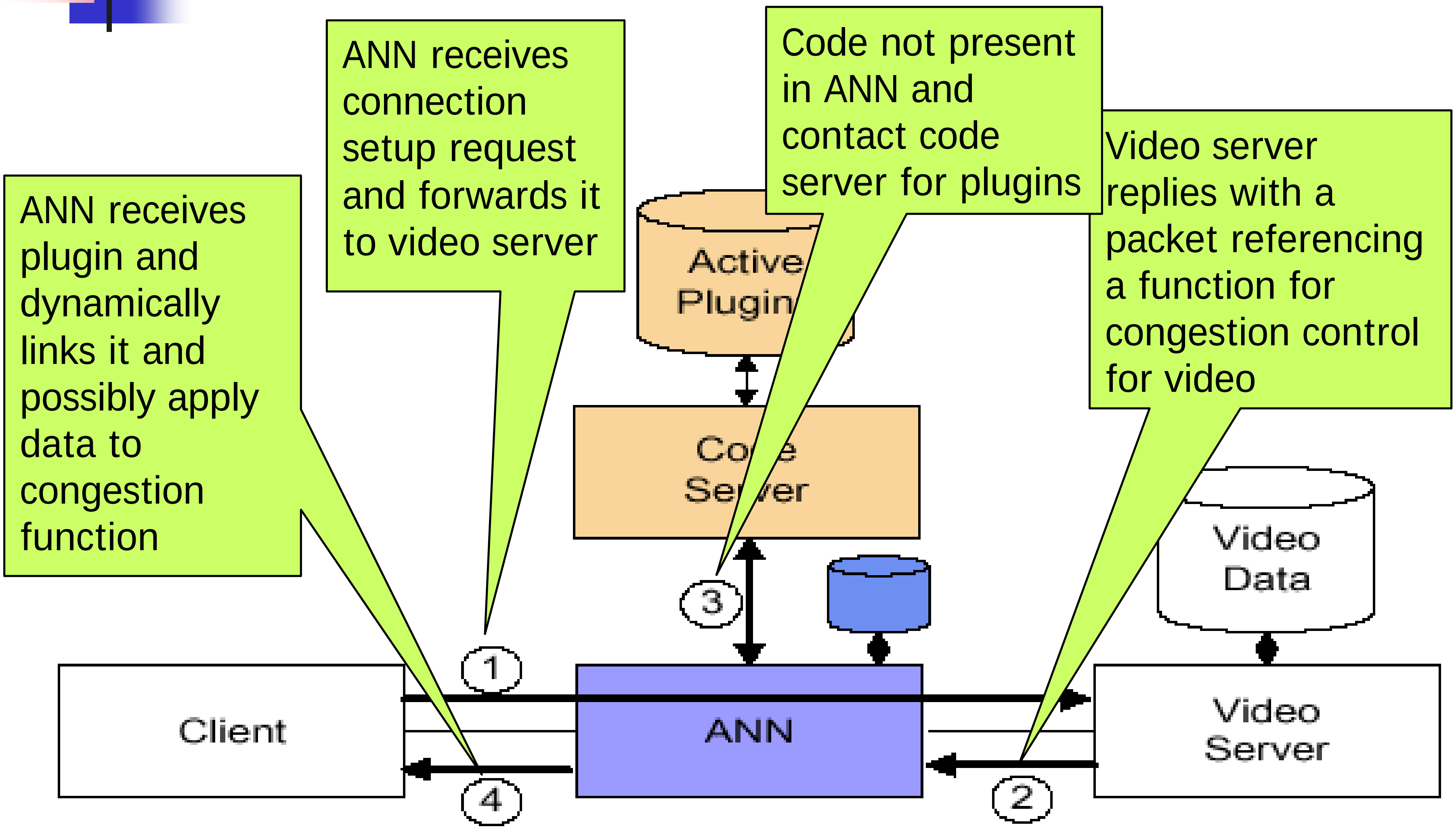
Distributed Code Cache for Active Network (DAN)

- A combination of capsule and programmable switch solution
- Capsule code => Reference to active plugins
- Reference of unknown code is downloaded from a code server
- Code fragments are dynamically linked and executes like native code on node
- Cryptography techniques are used for security to replace slow VM

DAN Mechanics

- Packets consist of sequence of IDs of functions and input parameters
- Demultiplexing packet to obtain the unique identifier (e.g. 0x0800 for IPV4)
- Hardware interface where packet arrive determines first function
- Last function (set) implemented in application
- Function ID act as pointer to code fragment
- Each function has option of not calling the next function
  - E.g. Forward packet to next hop w/o error detection
- Code fragments (plugins) are acquired from Code Server

Code Downloading Example



Security

- Active plugins are digitally signed by developers.
- Code Server serves as trusted, well-known node for the plugins and send authenticates plugin before sending.
- ANN only stores authenticated plugins.
- ANN can check plugin sources and developer before installation and running.

Minimizing Code Download Delay

- Download only happens once on first occurrence of a new function ID
- Probe Packet – Sent from server and down the packet path to each node to initiate download at each node (Parallelism).
- Optimal Code Server arrangement – Code servers should be close to ANN by using hierarchy similar to DNS servers.
- Administrator to select or find unicast address of a code server.
- Data server can also maintain a database of active plugins.

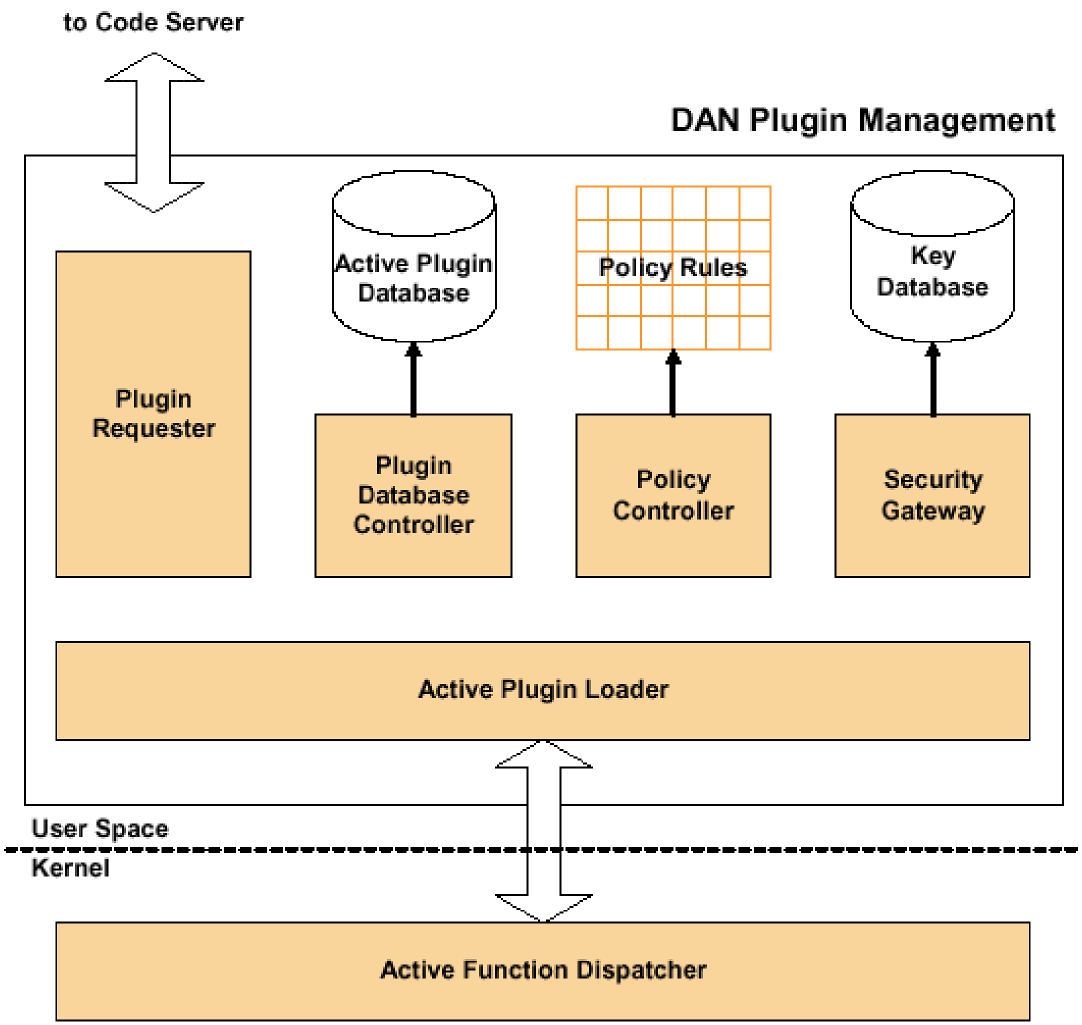
User Level Policies

- Acceptance Policy
  - Administrator is allowed to deny plugin even if plugin is trusted.
- Caching Policy
  - Timeout duration should be set by administrator to decide when a plugin expires.
  - Timeout can be set to infinite for non-expirable plugin.
  - Possibly treakable caching scheme (?)

Integration

- Integration by inserting new function ID at different layer.
- Data Link Layer
  - Use Link Layer Control SNAP field
- Network Layer
  - Use IP option, IPv6 option field is more extensible than IPv4
- Transport Layer
  - Function ID to replace or in addition to TCP/UDP function ID

DAN Execution Environment



DAN EE Components

- Active Function Dispatcher (AFD)
  - Scan packet for function ID
  - DAN function ID embed in ANEP header (or many other places)
  - Keeps track of all known function ID and a pointer to corresponding instances per flow.
  - Packet with a selector passed directly to first instance and does not go through AFD.
  - For unknown function ID
    - contacts Active Plugin Loader
    - Enqueue the packet in a dedicated queue
    - Proceed with next packet

DAN EE Components

- Active Plugin Loader (APL)
  - Dedicated kernel subsystem socket interface (like routed)
  - Talks to Policy Controller first and then possible contacts the plugin database
  - If plugin locally available, loads into subsystem
  - If not, contacts plugin requester to send request to code server.
  - If received plugin is verified and valid at the Security Gateway, loads into subsystem and stored by Plugin Database Controller.

DAN EE Components

- Policy Controller (PC)
  - Maintain policy as previously described.
- Security Gateway (SG)
  - Maintains a database of public keys
  - Implements full RSA public-key encryption using RSAREF library.
  - Library provides MD5 one way hash as RSA public key encryption.
    - ANN receives $\{\{Plugin\}_{MD5}\}_{Priv.Deve}$ and {Plugin}
    - Gets Pub.Deve and get $\{Plugin\}_{MD5}$ and calculates MD5({Plugin}) and see if
    - $\{Plugin\}_{MD5}$ =? MD5({Plugin})
- Security extension to DNS can be avoid by using IP address directly(?)
- Or can use IPv6's IP security
- Or associate DNS by public/private key scheme.

DAN EE Components

- Plugin Database Controller (PDC)
  - Administers local database for active plugins.
  - Database can be any database architecture and controller will be implemented to interface.
  - Store all plugin information, including keys.
- Plugin Requester (PR)
  - Responsible for requesting plugins and replying.
  - Request can be multicast, unicast or anycast.
  - And of TCP/UDP.
  - When using UDP, loss of active plugin request will cause those requesting active packet to drop.

Other Components

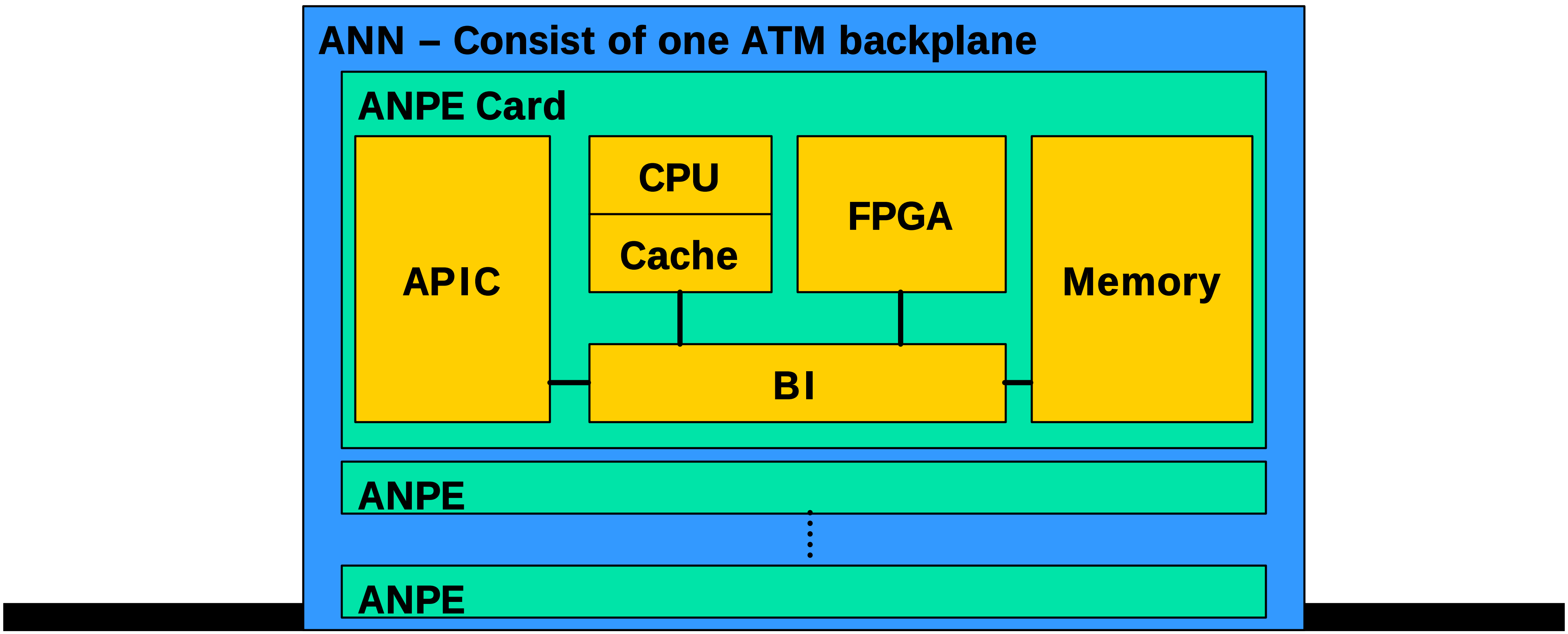
- Code Server
  - Can be of any architecture as long as being able to interface with requesting Plugin Requesters.
- Plugin Packages, contains
  - Code for 1+ active functions
  - Developer’s digital signature
  - Code server authentication information
  - Configuration information, e.g. Expiration time

Conclusion

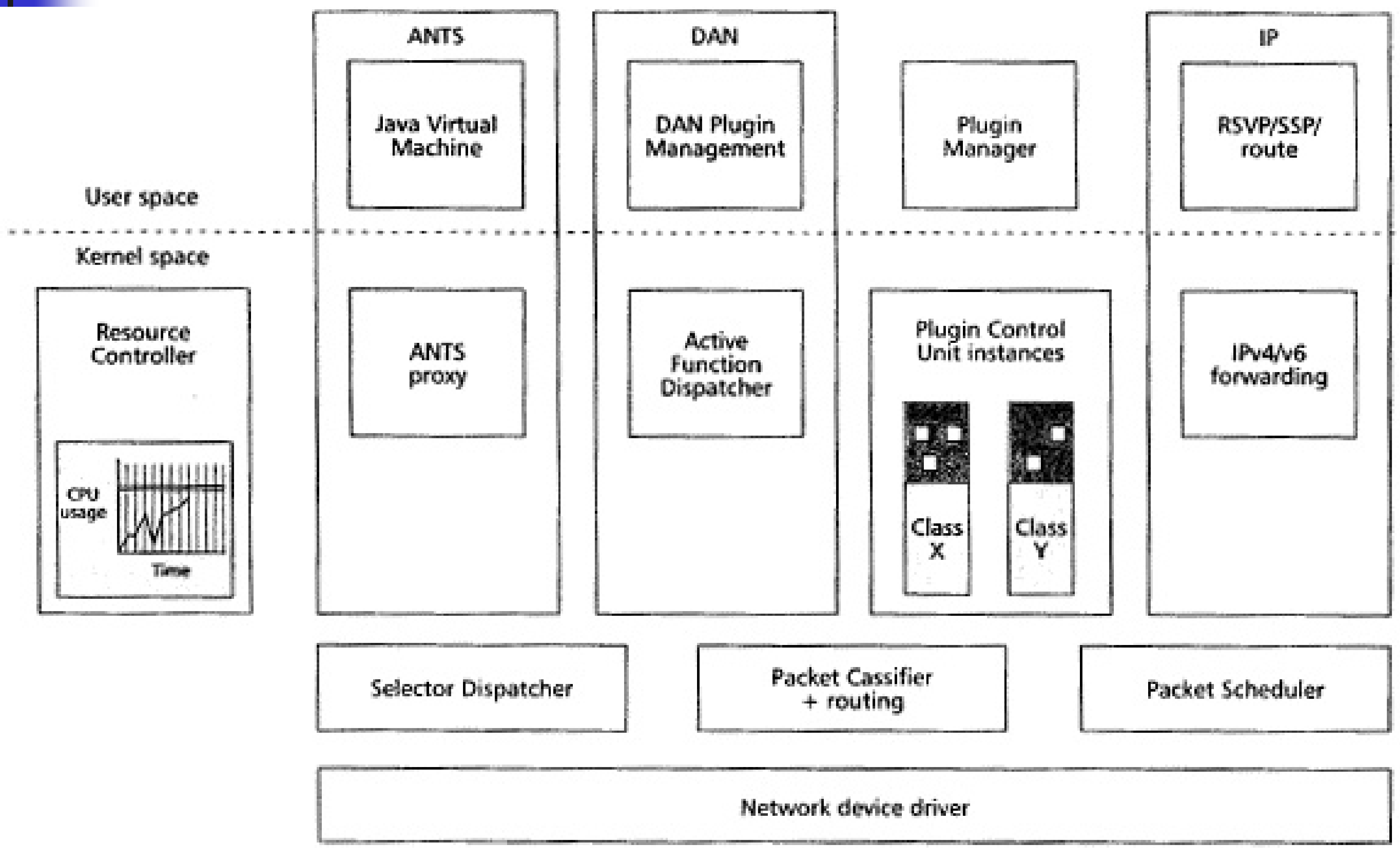
- This paper proposed a complete hardware, OS and to software architectual basis for an ANN.
- All parts are designed with scalablility in mind.
- Packets and plugins code fragments of most fundamental parts allowing application to be flexible and most importantly, scalable with the underlying hardware and OS architecture.

END

Hardware for ANN Overview



ANN Software Infrastructure



Object Daisy Chain