

Virtual Active Networks

Gong Su
Mar. 22, 2000

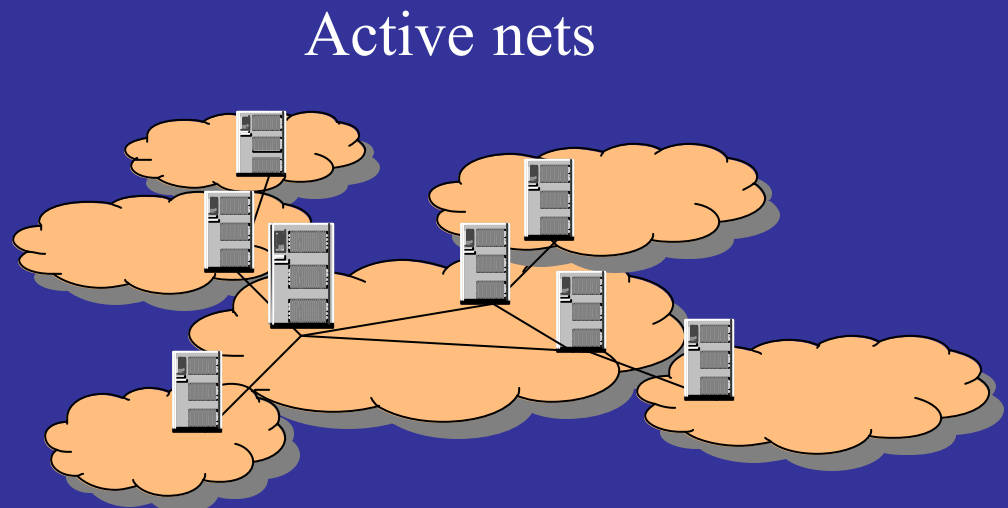
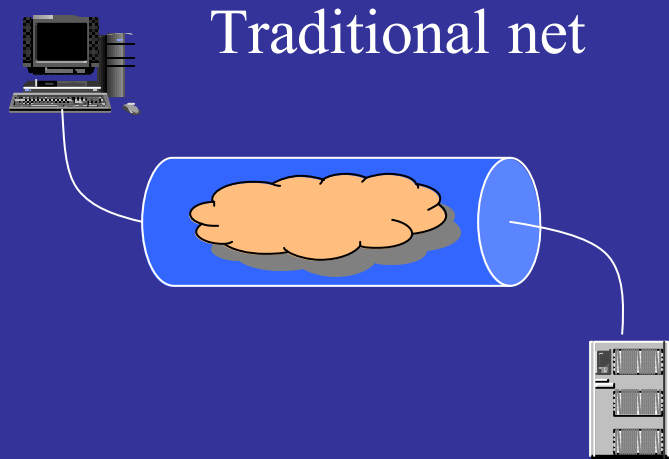
Multi-Edged Network Applications

☛ Traditional net apps: end-end computing

- Client-server software at end nodes
- The network is a best-effort packet-transport wire

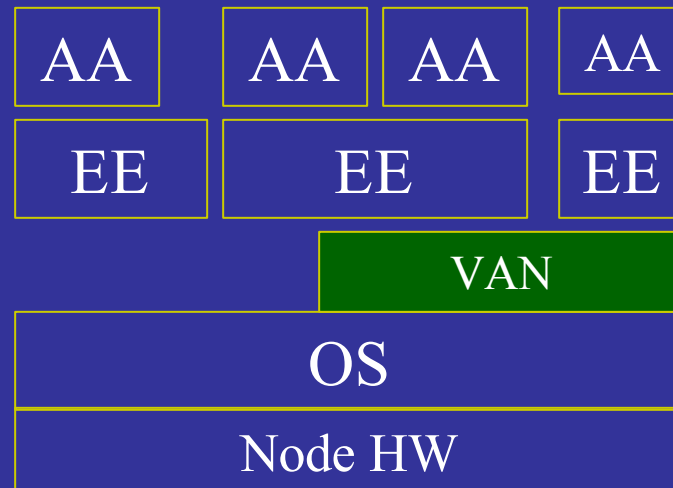
☛ Active net apps: multi-edge computing

- Application components dynamically deployed at net nodes
- Examples: intrusion protection, distributed simulation
- Need to interact with network resources & topology

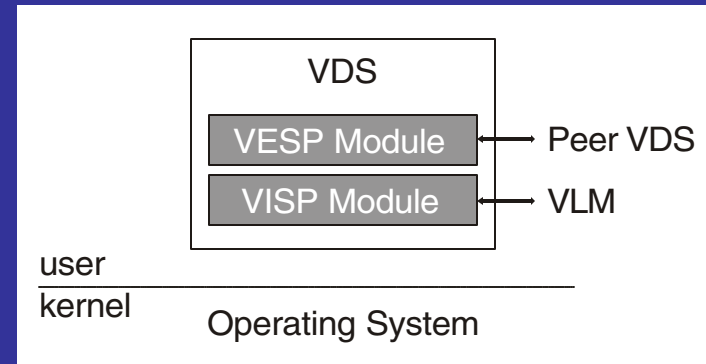
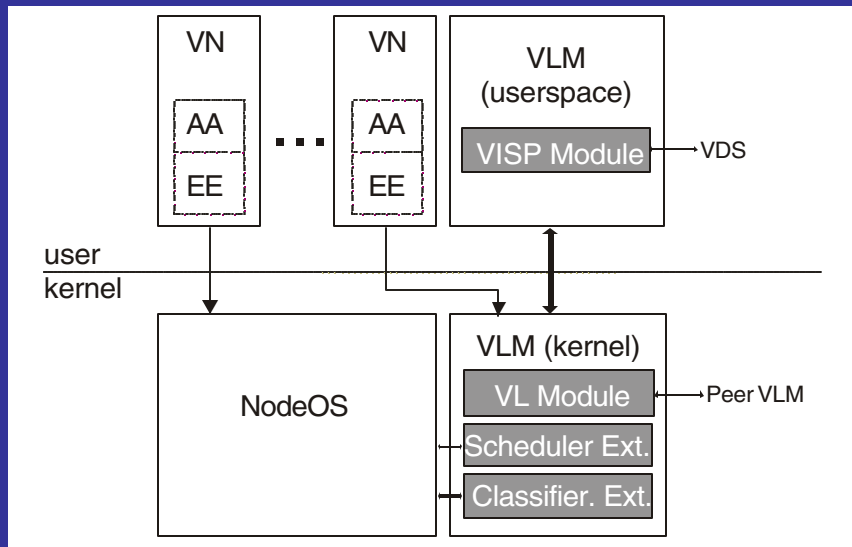


VAN Node Architecture

- Enables EE to configure net resources & topology
- Enables EE to monitor & adapt to net changes



VLM and VDS Functions



◆ VLM exports the following services to VDS'es and EE/AAs

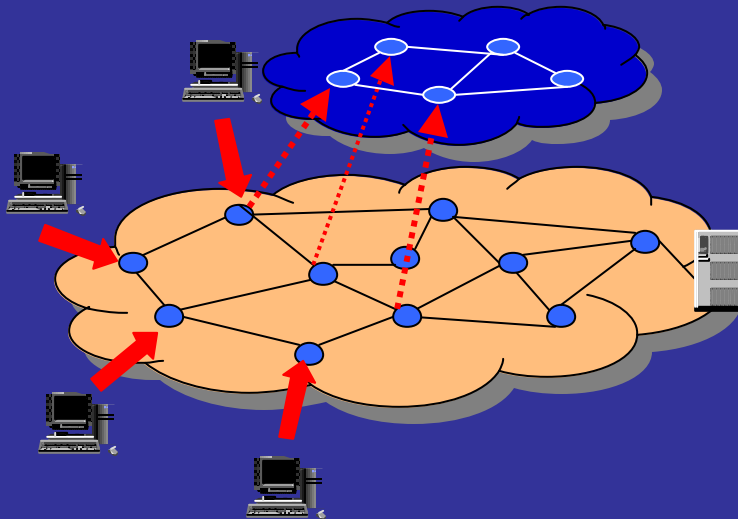
- create/delete VN with desired CPU resource
- create/delete VL with desired link resource
- modify VN and VL resources
- create/delete Virtual Ports (VPs)
- bind/unbind VPs to/from VLs
- send/receive messages to/from VPs

◆ VDS export the following services to EE/AAs

- create/delete a VAN
- join/leave a VAN

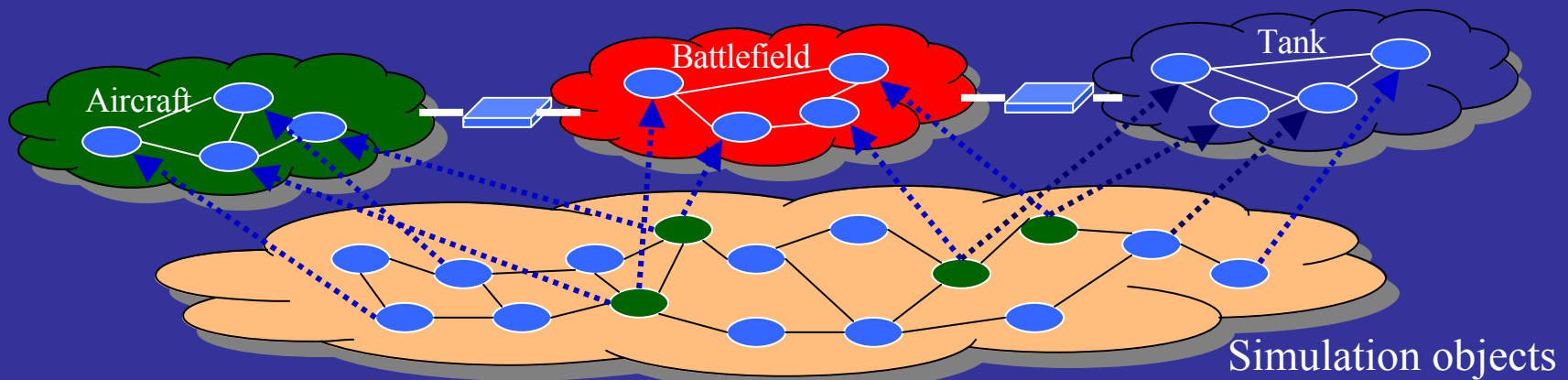
Using VAN For Active Fencing

- ☞ Idea: route suspicious traffic to a protection VAN
 - Create a VAN that captures traffic from suspicious sources
 - Deploy active filters to analyze traffic and choke attack
 - Example: choking denial of service attack using an active fence
- ☞ Active fencing as a protection paradigm
 - Dynamic: deploy response filters dynamically
 - Source-centric: fence the source rather than targets
 - Globally-coordinated protection



VAN Support of Active Simulation

- Problem: partition events flows in large-scale simulation
 - Event streams tend to cluster dynamically
 - Events are clustered by meaning, scenario & time scales
 - Need a way to allocate resources to serve the intensity of a cluster
- VAN is used to encapsulate clustered interactions
 - VAN support cluster interactions:
 - ➔ e.g., VAN of aircraft, VAN of tanks; VAN of a battlefield
 - Dynamic formation of VAN & dynamic allocation of resources
 - ➔ E.g., allocate more resources to battlefield VAN



Current Implementation

☛ VAN service APIs

- VAN creation/deletion
- VN, VP, and VL creation/deletion
- VN engine and VL engine loading/unloading and configuration
- Monitoring and event handling

☛ Experimental VN Engines and VL Engines

- VN engines: traffic redirection, IP routing (RIP capable)
- VL engines: plain UDP tunnel, QoS guaranteed (via RSVP)
UDP tunnel

☛ Command line front-end tool with scripting capability

☛ VAN MIB instrumentation – monitoring via SNMP

Recent Accomplishments

- VAN release 0.1
- QoS enabled Virtual Links
- Integration with ISI ASP
- Integration with UCLA Panda

Near Term Goals: Deployment on ABone

☞ Integration with anetd

- Livio Ricciulli at Metanetworks
- Adapt vanad to work with anetd

☞ Deployment on ABone

- Bob Braden at ISI
- Make vanad part of ABone core software

Long Term Goals: Kernel Support & Applications

☞ Kernel resource management support

- Scheduler extension
- Classifier extension
- Virtual link setup (QoS signaling + tunnel encapsulation)

☞ Demo active applications

- Active fencing
- Others?