

Safe Execution of Untrusted Applications on Embedded Network Processors

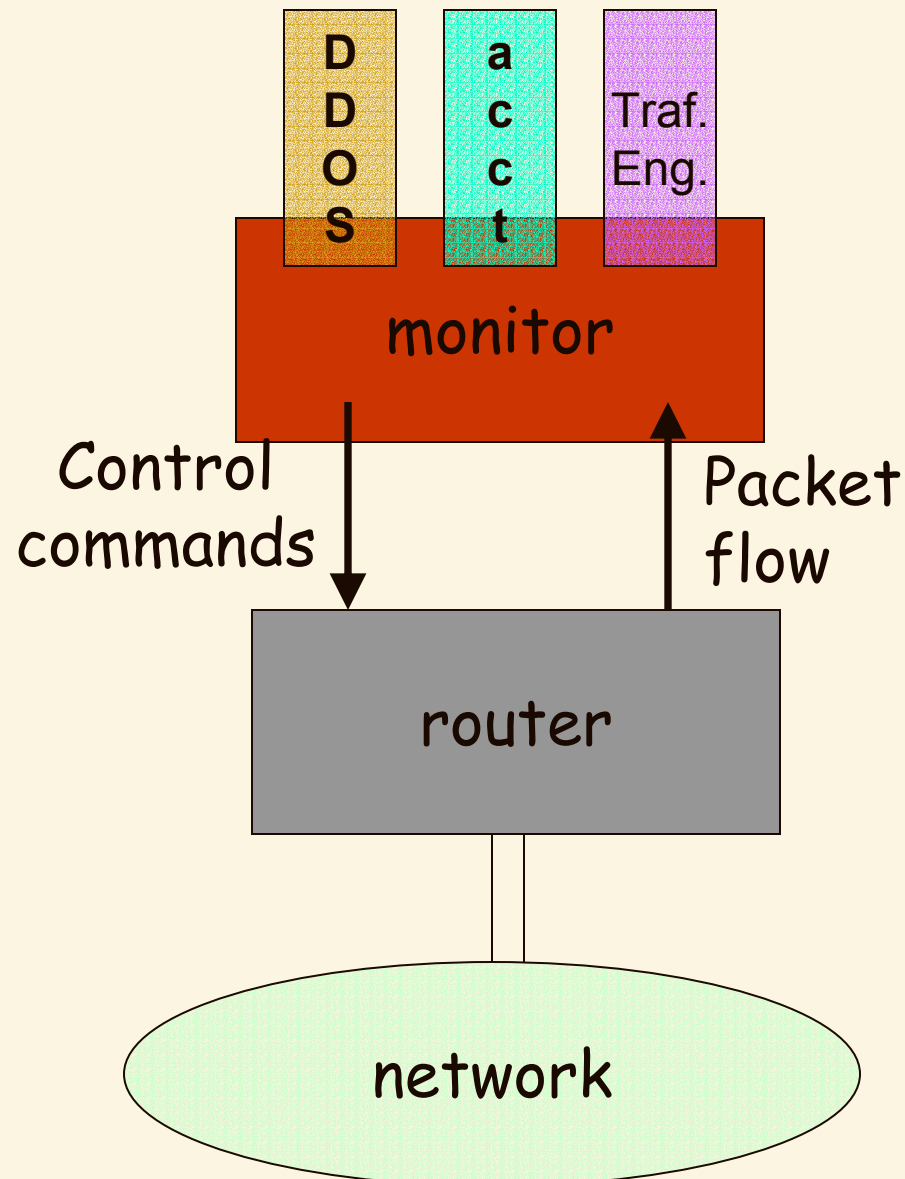
Kostas G. Anagnostakis - Penn

Overview

- **SPYCE goal:**
 - *"maintaining and managing a computational infrastructure, distributed among many heterogeneous nodes that do not trust each other completely ..."*
- **This presentation:**
 - **high-performance** systems for diffuse computing
 - "edge" routers, overlay nodes, embedded
 - Extending **FLAME** software architecture to Intel **IXP** network processor hardware

FLAME high-level architecture

(joint work w/ Ioannidis, Greenwald, Miltchev, Smith)



FLAME goals & principles

- **Goals:**

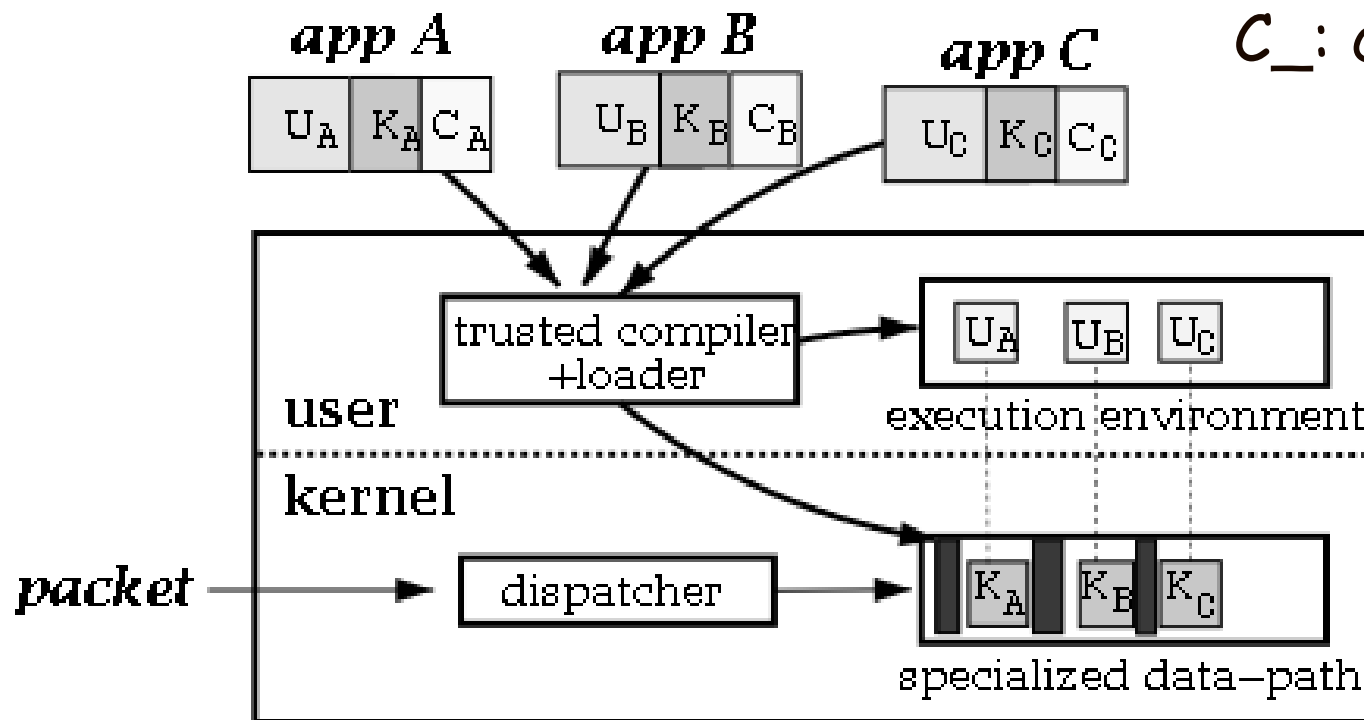
- Executing untrusted network monitoring code on high speed links (Gbit/s)
- Current focus on **passive packet processing**, but design can be adapted for other types of network functions

- **Principles:**

- **Off-the-shelf** hardware and software
- Efficient **low-level packet handling** and **language-based protection**
- Ease of programming through **Cyclone**, type-safe C dialect
- Fine-grained policy control through **Trust Management**
- Software-based fault isolation (SFI) for controlling memory accesses + processing time

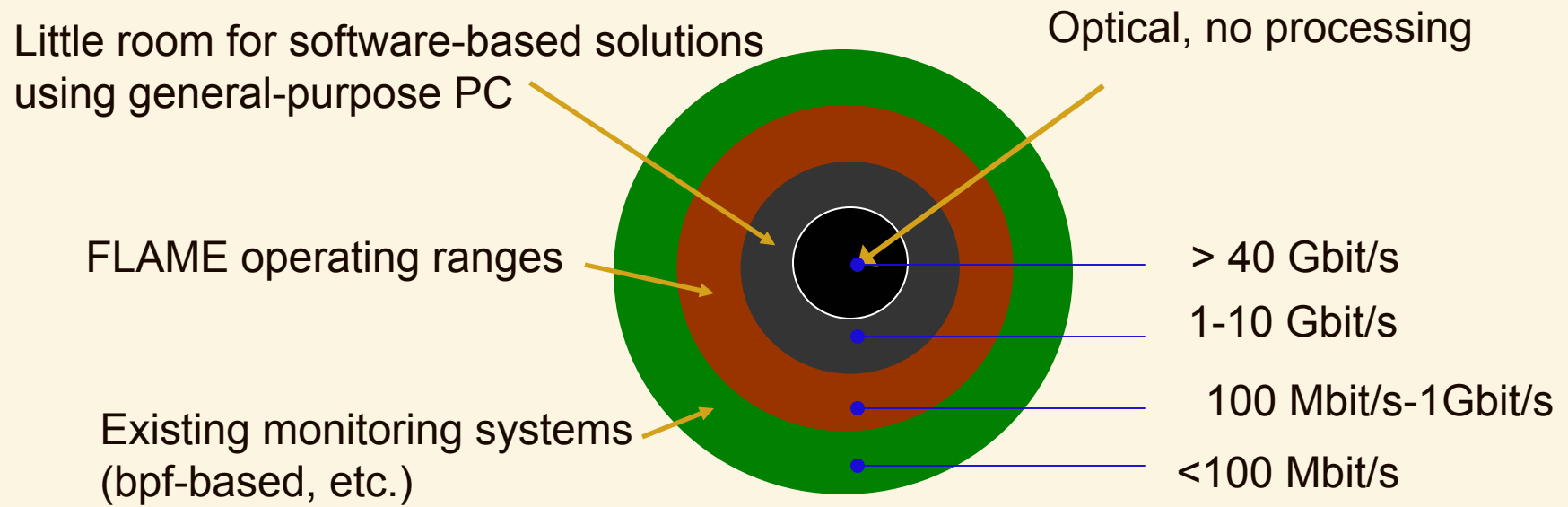
FLAME software architecture (PC)

U_: user-level code
K_: kernel-level code
C_: credentials



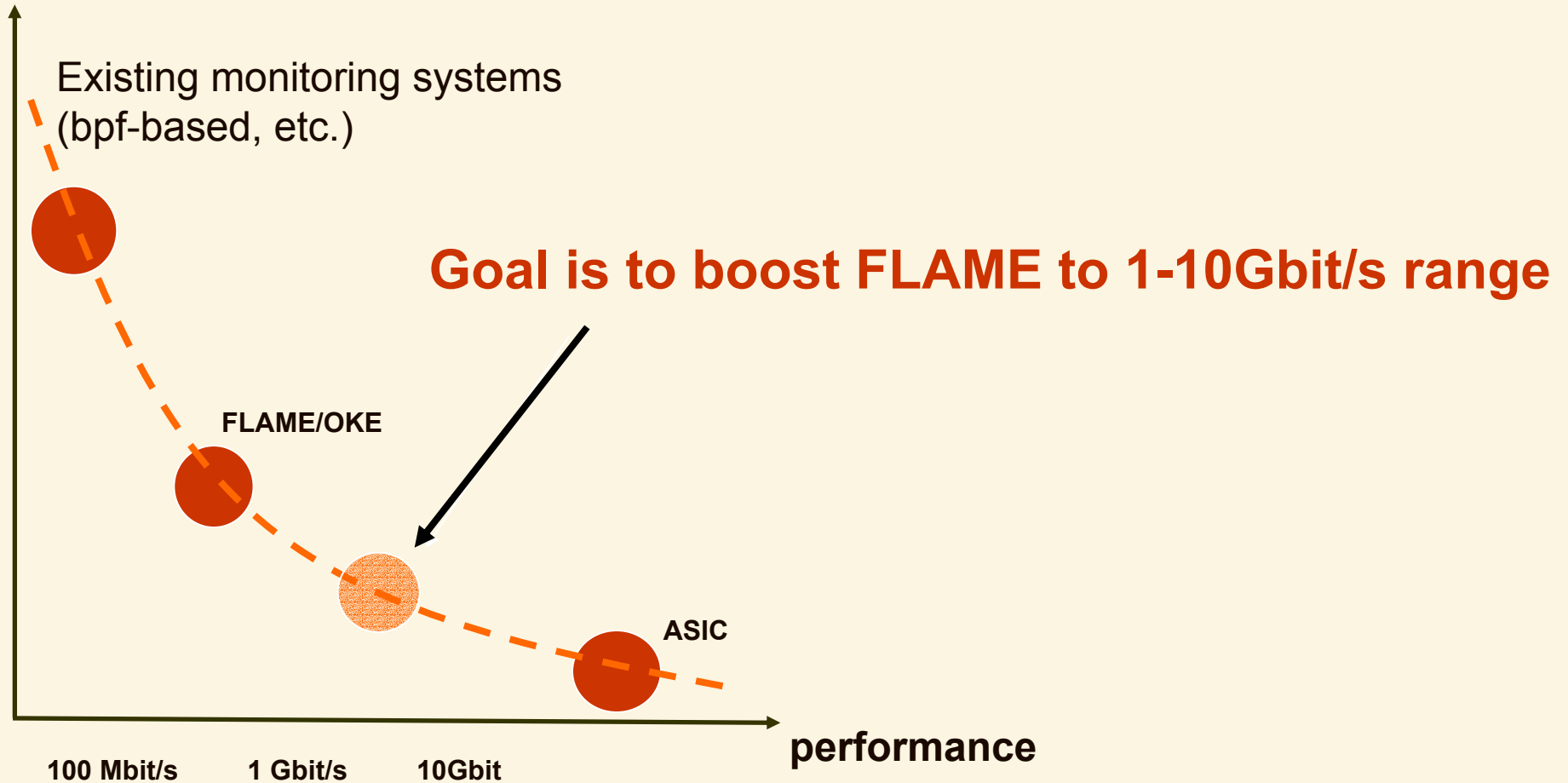
Monitoring System

Performance issues



Performance vs. flexibility

flexibility



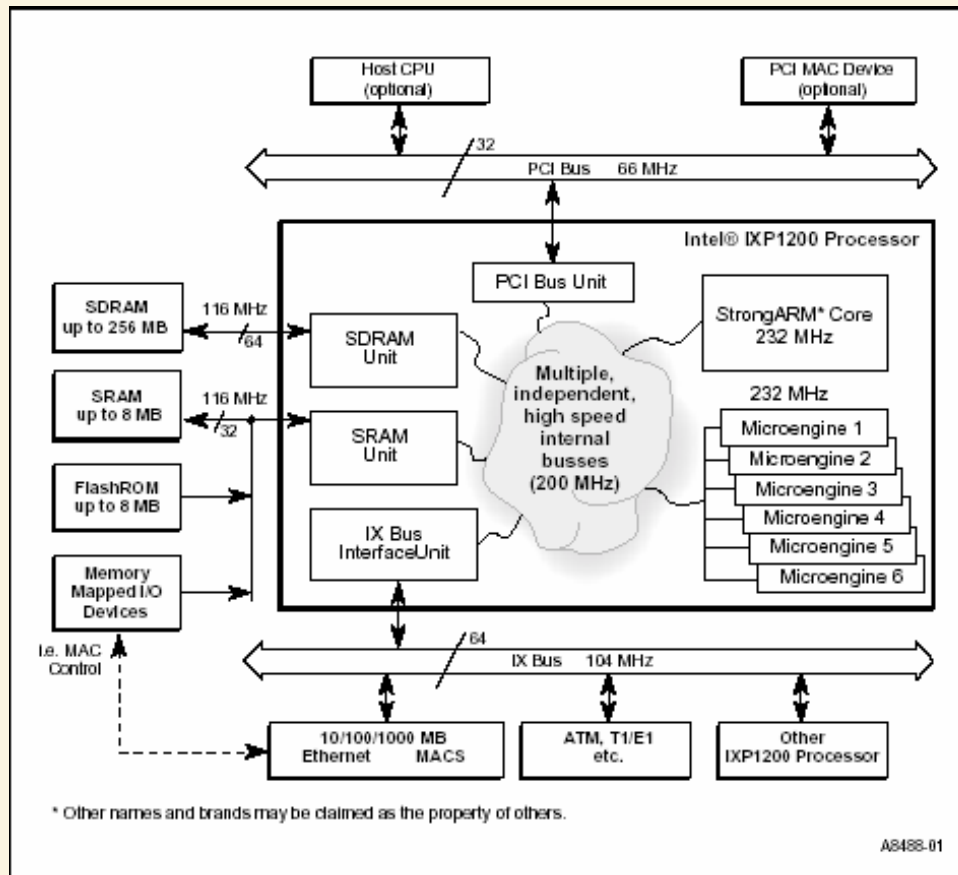
Network Processors

Designed for high-performance networking functions:

- extensive parallelism
- hardware-assists (hashing, checksums, ...)
- high-speed buses

Main problem:

- no support for familiar multiprogramming model
- complex architecture
- no OS, MMU, ...



IXP 1200 Network Processor

Mapping FLAME on the IXP NP

(joint work w/ Bos, Cristea, Samwel – LIACS)

- **Packet reception and control:**
 - dispatcher on one microengine
 - single-producer/multiple-consumer buffer mgmt.
- **Processing bounds:**
 - each application allocates one or more microengines
 - dispatcher enforces processing bounds, no SFI needed
- **Memory access safety:**
 - SFI only necessary on shared memory
 - memory abstraction hard, code needs to manage different types of memory itself
- **SFI protects code utilizing other resources**
 - For example: memory bus, PCI

Experimental evaluation

- Proof-of-concept prototype:
 - **Diet-OKE** based on LIACS **OKE** implementation
- Maximum packet processing rate:
 - **~1M pkts/sec** , 64-byte packets (**600 Mbit/s**)
 - **~60K pkts/sec** , 1500-byte packets (**700 Mbit/s**)
- Cost of safety:
 - Workload: content matching, diffserv marking, SYN detection, RPC scan
 - Results: overhead in 15-30% range

Potential applications

- **Large-scale instrumentation:**
 - **network tomography** becomes trivial
- **Emergency response:**
 - Rapid deployment of **virus blocking, quarantine** functions
- **Application boosters:**
 - On-demand **transcoding**
- **Smart transport:**
 - Overcoming problems with **wireless, multiple congestion points**