

CSE 58x: Networking Practicum

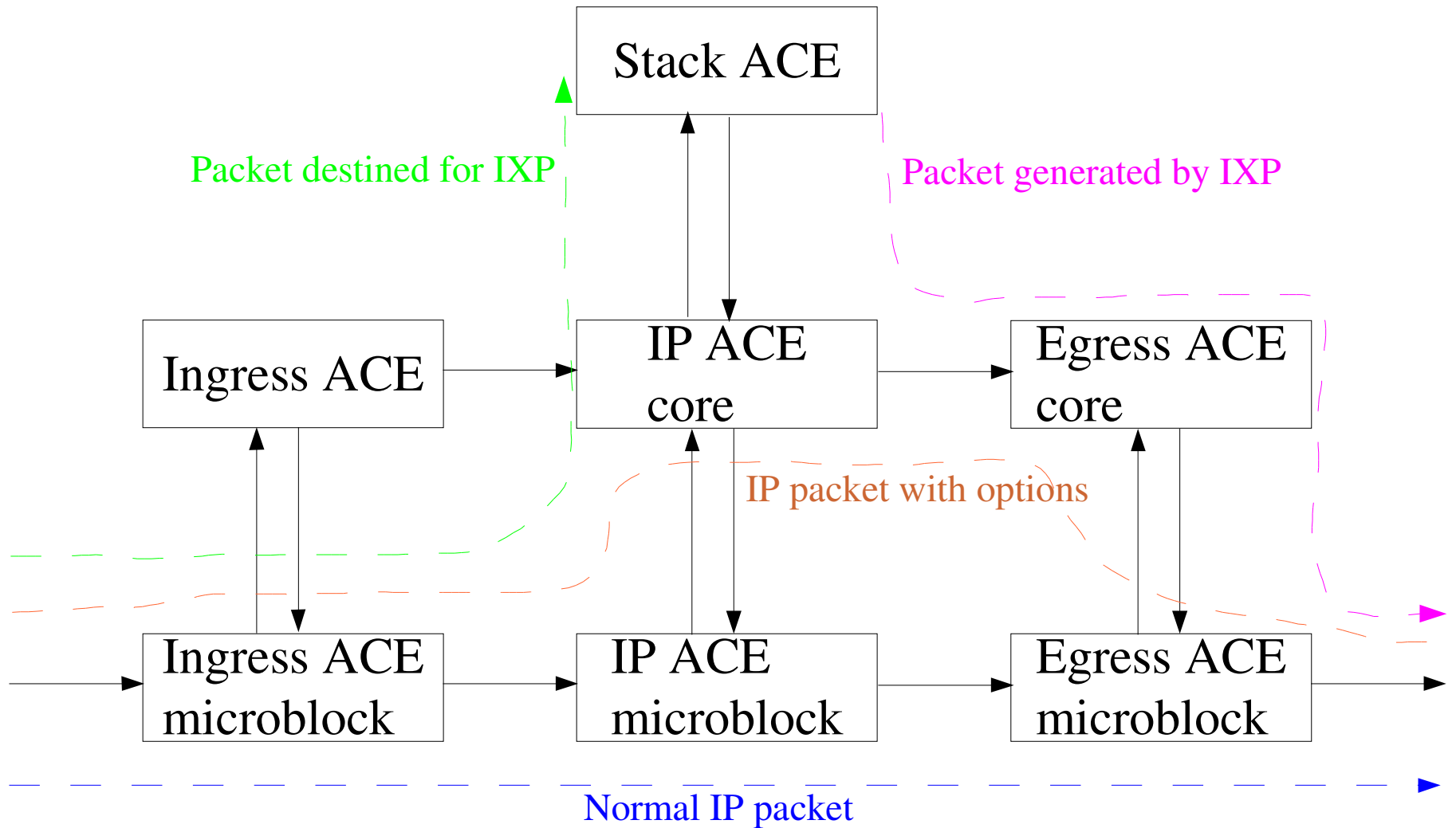
Learning the ACE Framework
Lecture 3

ACE framework

- Active Computing Element
 - Fundamental software building block
 - Composed to implement full function (both slow and fast path)
 - MicroACEs
 - Core component => StrongARM (written in C)
 - `ixasdk/src/microace/aces/tutorial1/count_ace1/source`
 - Microblock component => micro-engines (written in assembly)
 - `ixasdk/src/microace/aces/tutorial1/count_ace1/microblock`



Example



Microblocks

- Rely on StrongARM to allocate memory (See init.c)
- Two components
 - Initialization macro
 - Initializes data structures
 - Processing macro
 - Does the work
- Microblock groups
 - Specify the set of microblocks that run on a single micro-engine

Dispatch loop

- Controls packet flow in between microblocks of a microblock group
 - Infinite loop that invokes microblocks
 - Return code from microblocks determine next microblock to execute (`dl_next_block`)
 - Shared buffer handle passed to/from microblocks of the group
 - Unidirectional inter-engine packet queues to pass packets between microblocks residing on different micro-engines
 - Intel's DL buffer management macros (p. 399)
 - `ixasdk/src/microace/common/dispatch_loops`

Some dispatch loop macros

- DL_SASink, DL_SASource
 - Queues to and from StrongARM (core components)
 - DL_SetAceTag to demux packet based on ACE
 - DL_SetExceptionCode to identify code
- EthernetIngress, DL_EgressSink
 - Queues to/from Ingress and Egress
 - `ixasdk/src/microace/common/dispatch_loops/Count1_Ingress`
 - `ixasdk/src/microace/common/dispatch_loops/Egress_{Fifo,RoundRobin}`

Some dispatch loop macros

- DL_MESink
 - Queue to another microblock group
 - Binding specified in ixsys.config
- DL_Drop
 - Drop packet, recycle buffer/memory resources
- DL_GetBufferOffset, Buf_GetData
 - Packet header held in pre-allocated transfer registers (xbuf_alloc)
 - Buf_GetData gets location of packet buffer
 - DL_GetBufferOffset gets offset of start of packet within

Dispatch loop and exceptions

- IX_EXCEPTION code
 - Reserved for packets that require processing in slow path (StrongARM)
 - Packet placed onto queue that leads to the StrongARM
- Information passed to StrongARM
 - ACE Tag
 - Each ACE assigned a unique identification number
 - Tag used to identify which core component should handle exception
 - Exception code
 - Each core component provides exception handler per code

Microengine assembly syntax

- label: operator operands token
- Assembler directives
 - Line begins with a “.”
 - For symbolic aliasing, declarations, structured code
- Macro preprocessor directives
 - Line begins with a “#”
 - File includes, substitutions, code expansion
- Registers, memory, instructions, context switching
 - Read Chapter 24