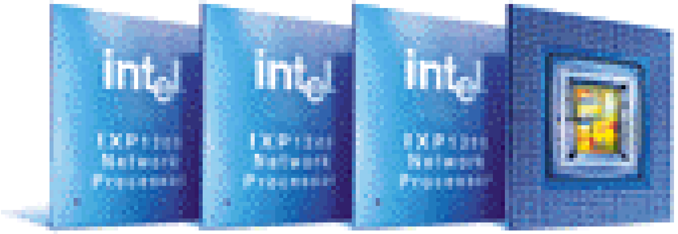
Embedded Software in Network Processors – Models and Algorithms

By Lothar Thiele, Samarjit Chakraborty, Matthias Gries, Alexander Maxiaguine, and Jonas Greutert

Presented by Doug Densmore
Will Plishker
April 2, 2002

Outline

- Introduction
- Motivation
- Model Computation for Packet Processing
- Modeling Discrete Event Streams and Systems
- Task Scheduling in Network Processor
- Design Space Exploration
- Concluding Remarks



Network Processors (NP)

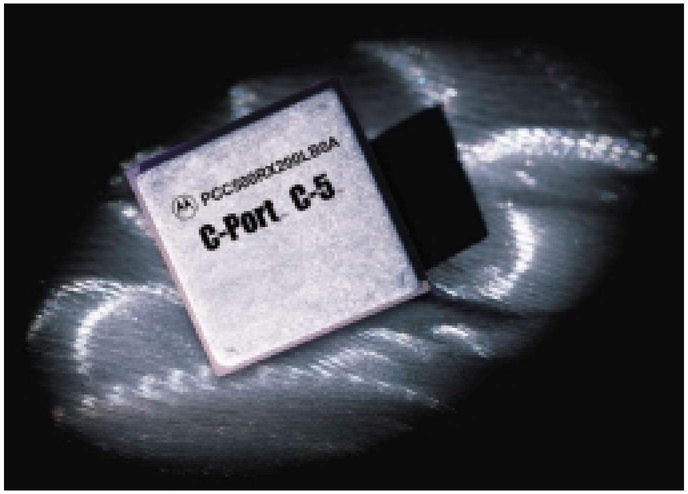
- Highly programmable, dedicated processors optimized to perform packet processing functions.
- Two basic tasks
  - Packet processing
  - Traffic Management

Network Processors (NP)

- Architecture and implementation depend very much on its placement in the network hierarchy.
- Access Network Level
  - Support a wide and varied range of packet processing. Relatively low data rates.
- Core/Backbone Network Level
  - High data rates but restricted processing capabilities.

Examples of Network Processors

- Intel IXP1200 Family
- Motorola C-Port C-5
- Lexra NetVortex
- Agere PayloadPlus
- Niraj Shah
  - "Understanding Network Processors"



Motivation

- Due to the highly programmable nature, software is an integral part of an NP.
- Papers have proposed different software architectures for flexible configurable routers.
- However, there has been no formal and unified study of this subject.
- Need a formal study of packet processing devices!!

Motivation

- Framework based on models used in packet scheduling
  - Task and Resource Model for NP
  - Calculus of packet streams and processing
- Paper will consider two examples:
  - Task scheduling in an embedded processor
  - Hardware/Software interactions

Model of Computation for Packet Processing

- Definition 1 – Task Structure
  - Set of flows $f \in F$
  - Set of tasks $t \in T$
  - Connected, directed, and acyclic task graph G(f), for each flow f.
  - G(f) consists of a set of task nodes T(f) and a set of directed edges E(f).
  - G(f) has a unique source node s(f).

Task Graph

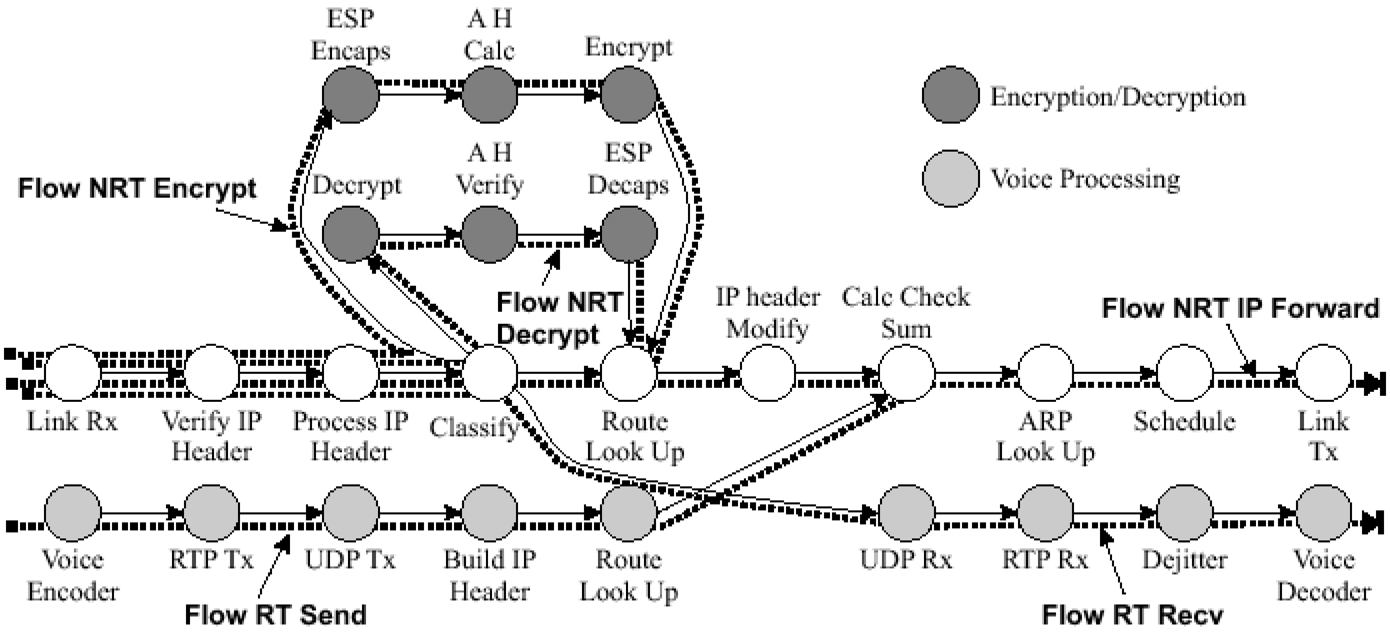


Fig. 1. Example of a task graph corresponding to a simple network processor, see Definition 1.

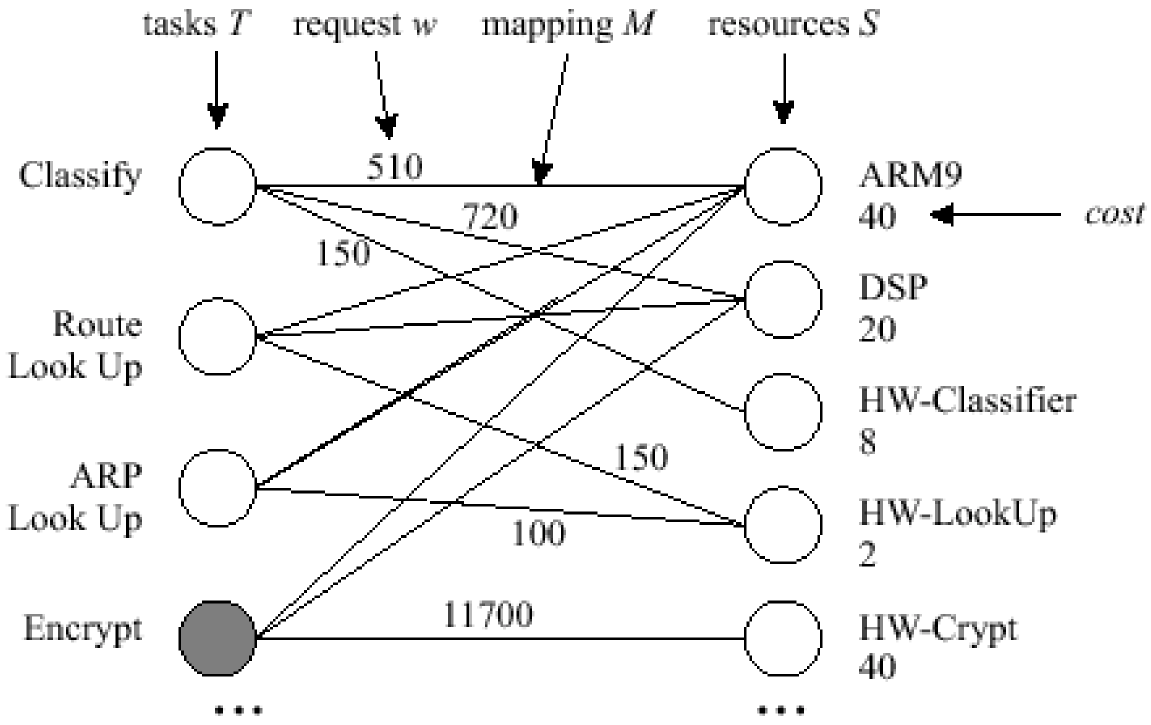
Model of Computation for Packet Processing

- Definition 2 – Resource Structure
  - Set of resources $s \in S$
  - $S \rightarrow \mathbf{R}^+$ = relative cost of a resource (i.e. power, area)
  - $M \subseteq T \times S$ defines the possible mapping of $t \in T$ to resources.

Model of Computation for Packet Processing

- Definition 3 – Timing Properties
  - To each flow f ∈ F there is an end to end deadline $d: F \rightarrow R^+$
  - If a task t can be executed on a resource s, then it creates a “request” w. $w(t,s) \in R^+$
  - This request can be thought of as a number of instructions.

Example of a resource structure



Modeling Discrete Event Streams and Systems

- Traditionally event streams are modeled statistically.
- Hard bounds are more appropriate modeled by discrete event streams and systems.
- Arrival and service curves are bound.

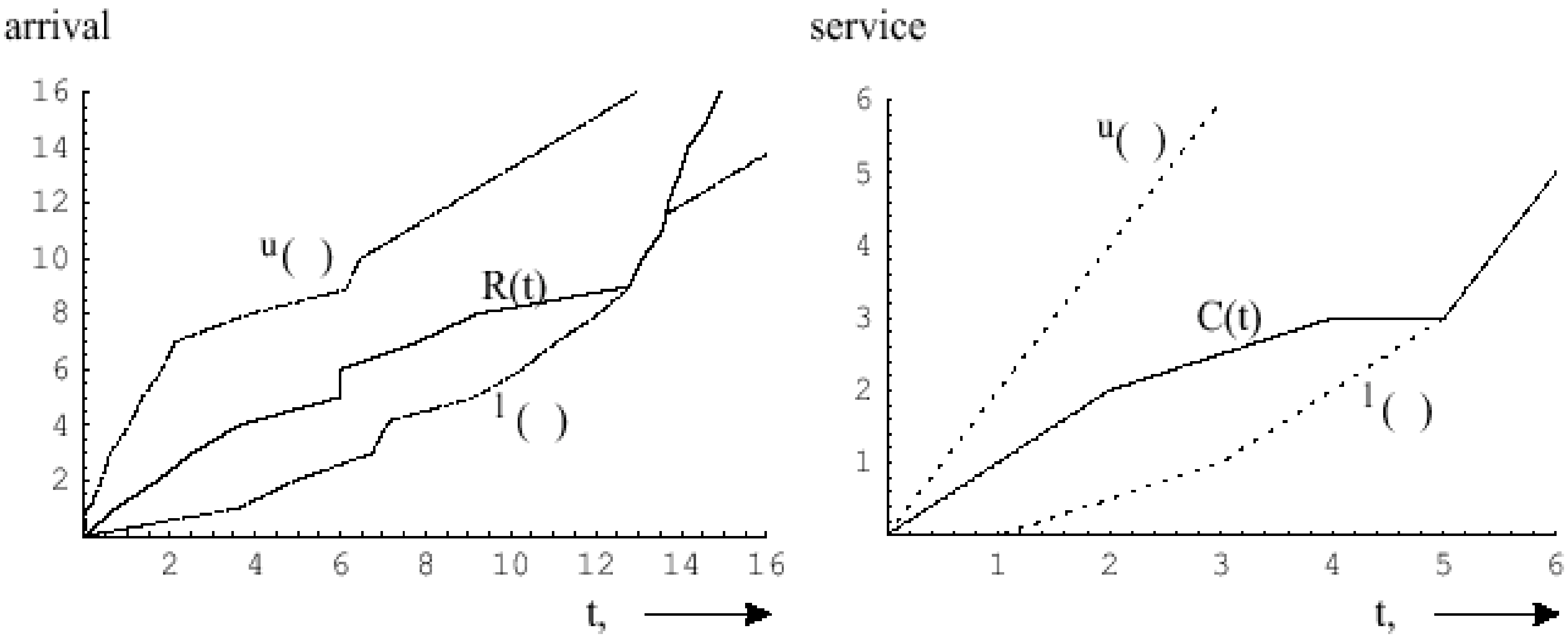
Modeling Discrete Event Streams and Systems

- Definition 4 – Arrival Service Function
  - Arrival function R(t), denotes the number of events that have arrived in the interval [0,t].
  - Service function C(t), denotes the number of events that could have been serviced in the interval [0,t].
  - Events may be packets, bytes, instructions, etc.
  - C(t) and R(t) are non-decreasing

Modeling Discrete Event Streams and Systems

- Definition 5 – Arrival and Service Curves
  - Two time instances s and t
  - $\Delta = t\text{-}s$
  - Upper arrival curve $\alpha^u(\Delta)$ and lower arrival curve $\alpha^l(\Delta)$
  - $\alpha^l(\Delta) \leq R(t)-R(s) \leq \alpha^u(\Delta)$
  - Upper service curve $\beta^l(\Delta)$ and lower service curve $\beta^u(\Delta)$
  - $\beta^l(\Delta) \leq C(t)-C(s) \leq \beta^u(\Delta)$

Arrival and Service Curves



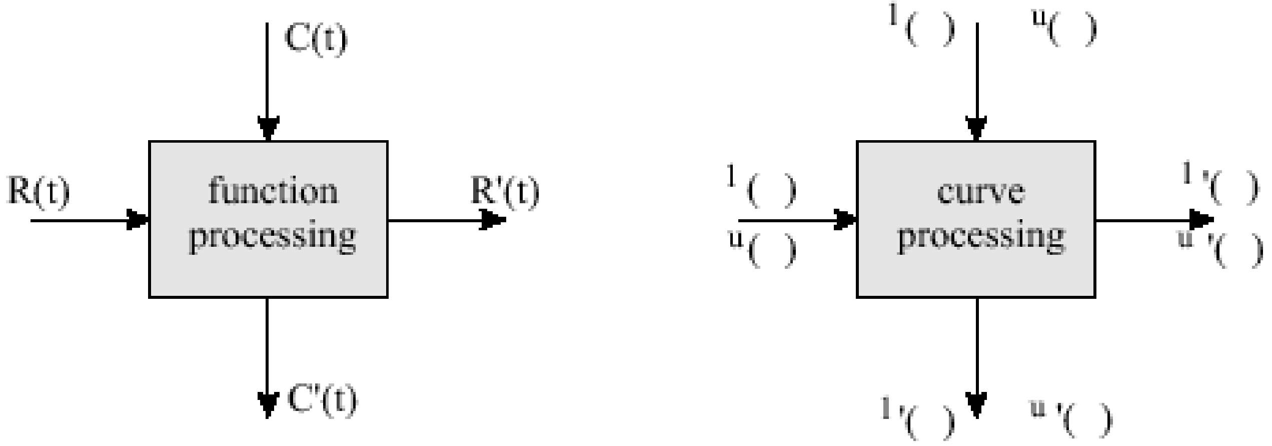
Modeling Discrete Event Streams and Systems

- Definition 6 – Curves and Flows
  - To each flow f there are associated upper and lower arrival curves.
  - To each resource s there are associated upper and lower service curves.

Modeling Discrete Event Streams and Systems

- Definition 7 – FUNction Processing
  - Given a resource node s with its corresponding service function C(t) and an event stream described by the arrival function R(t) being processed by s, we have:
  - $R'(t) = \min\{R(u) + C(t) - C(u)\}$
    - Amount of computation delivered to process event stream
  - $C'(t) = C(t) - R'(t)$
    - Remaining computation available
  - $0 \leq u \leq t$

Processing of event streams



Modeling Discrete Event Streams and Systems

- Proposition 1 – Curve Processing
  - Given an event stream described by the arrival curves $\alpha^l(\Delta)$ and $\alpha^u(\Delta)$ and a resource node described by the service curves $\beta^l(\Delta)$ and $\beta^u(\Delta)$, then the following expressions bound the remaining service function of the resource node and the arrival function of the processed event stream.

Modeling Discrete Event Streams and Systems

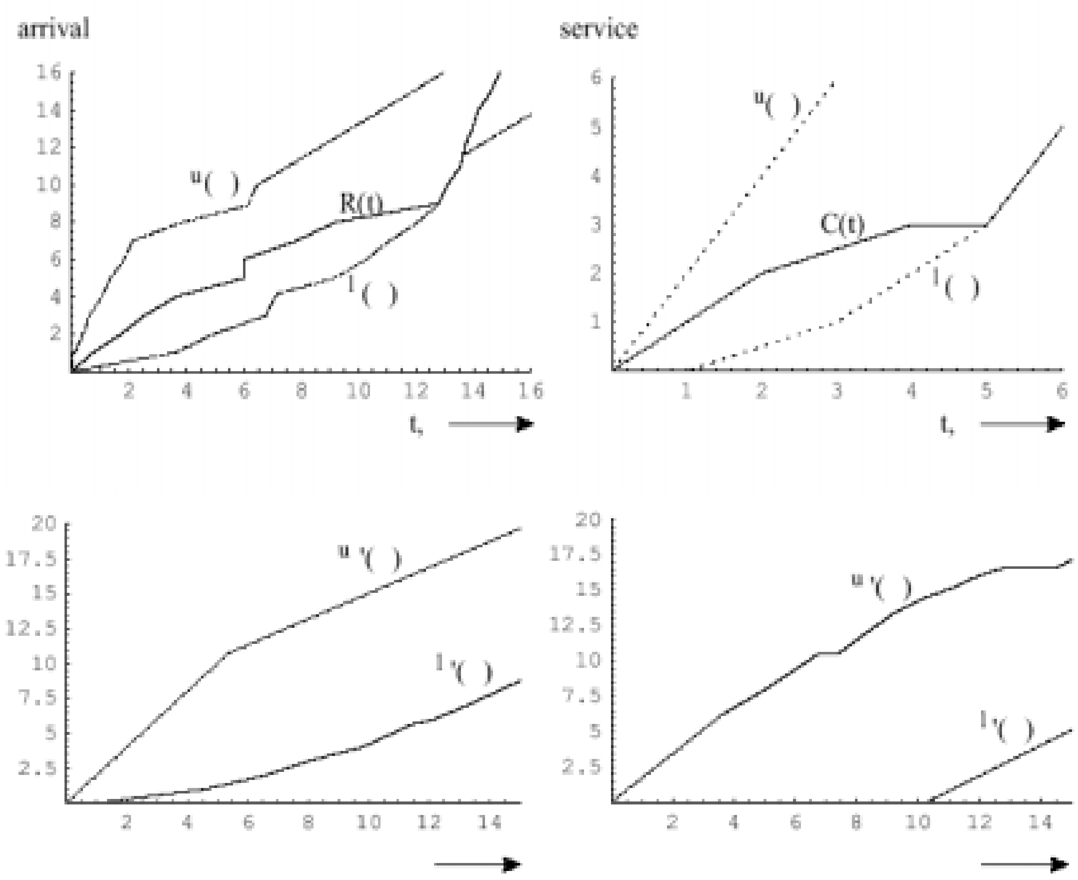
$$\alpha^{l\prime}(\Delta) = \min_{0 \le u \le \Delta} \{\alpha^{l}(u) + \beta^{l}(\Delta - u), \beta^{l}(\Delta)\}$$

$$\alpha^{u\prime}(\Delta) = \min_{0 \le u \le \Delta} \left\{ \max_{v \ge 0} \{\alpha^{u}(u+v) - \beta^{l}(v)\} + \beta^{u}(\Delta - u), \beta^{u}(\Delta) \right\}$$

$$\beta^{l\prime}(\Delta) = \max_{0 \le u \le \Delta} \{\beta^{l}(u) - \alpha^{u}(u)\}$$

$$\beta^{u\prime}(\Delta) = \max_{0 \le u \le \Delta} \{\beta^{u}(u) - \alpha^{l}(u)\}$$

Processing of the Curves



Simple Processing Network Example

- Set of flows, $f_1,\ldots,f_n$
- Associated event streams R1(t),…,Rn(t) ordered according to decreasing priority.
- Each flow, f_i must have a task t_i executed on one resource s with an associated request $w(t_i,s)$
- Arrival curves for flow f_i
- Service curves for resource node s

Simple Processing Network Example

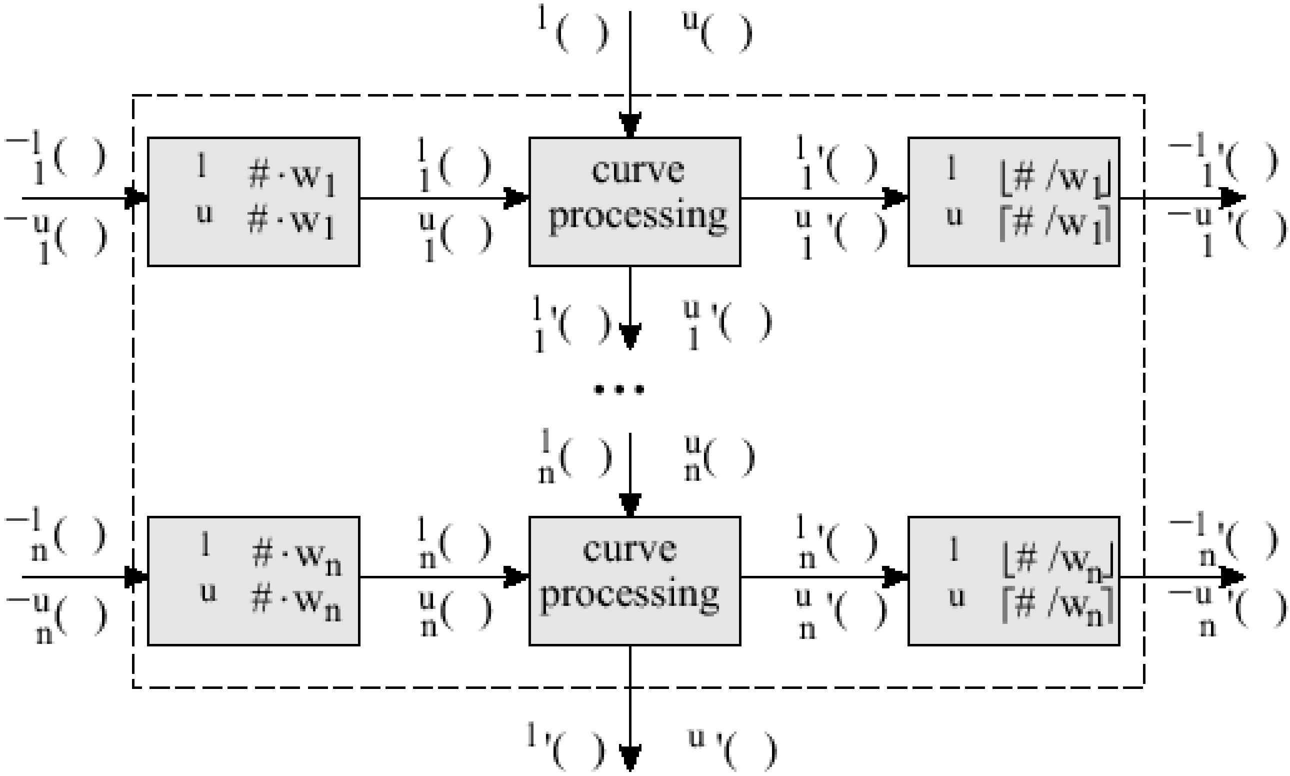
$$\alpha_i^u(\Delta) = w_i \cdot \overline{\alpha}_i^u(\Delta) \quad , \quad \alpha_i^l(\Delta) = w_i \cdot \overline{\alpha}_i^l(\Delta)$$

$$\overline{\alpha}_i^{u\prime}(\Delta) = \lceil \alpha_i^u(\Delta)/w_i \rceil \quad , \quad \overline{\alpha}_i^{l\prime}(\Delta) = \lfloor \alpha_i^l(\Delta)/w_i \rfloor$$

$$\beta_1^u(\Delta) = \beta^u(\Delta) \quad , \quad \beta_i^u(\Delta) = {\beta_{i-1}^u}'(\Delta) \quad \forall 1 < i \leq n \quad , \quad \beta^{u\prime}(\Delta) = \beta_n^{u\prime}(\Delta)$$

$$\beta_1^l(\Delta) = \beta^l(\Delta) \quad , \quad \beta_i^l(\Delta) = {\beta_{i-1}^l}'(\Delta) \quad \forall 1 < i \leq n \quad , \quad \beta^{l\prime}(\Delta) = {\beta_n^l}'(\Delta)$$

Diagram showing a processing network



Task Scheduling in Network Processors

- Problem: Schedule the CPU cycles of the processor to process a mix of real-time and non-real-time packets.
- All real time packets meet their deadlines.
- Non-real-time packets experience minimum processing delay.
- EDF based scheduling method

Task Scheduling in Network Processors

- Given flows F
  - Two disjoint subsets, F_{RT} and F_{NRT}
- All flows $f_i \in F_{RT}$ have deadlines $d(f_i)$
- Constrained by upper arrival curve α^u_i
- Processing cost of flow f_k on a single resource s is denoted by $w(f_k)$

Task Scheduling in Network Processors

- Flows in F_{NRT} have no time constraints
- Used to model packet streams corresponding to bulk data transfers such as FTP.
- Processing cost of each packet flow, f_k, on a single resource s is denoted by $w(f_k)$ – same as for F_{RT}

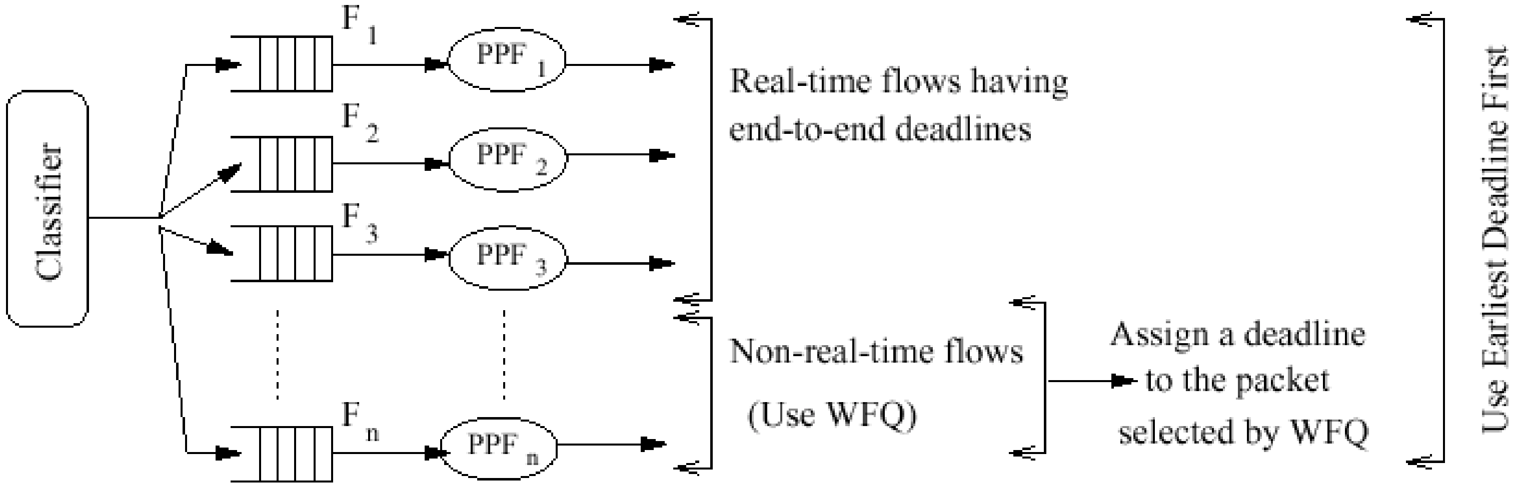
Task Scheduling in Network Processors

- Objective of this scheduling algorithm:
  - To guarantee that all real time packets meet their associated deadline.
  - That non-real-time packets experience the minimal possible delay.
  - Associate with each non-real-time flow f_j a weight ϕ_j, and use that weight to allocate CPU cycles.

Task Scheduling in Network Processors

- Hierarchical EDF
- Weighted Fair Queuing for non-real-time flows based on Generalized Processor Sharing algorithm.
  - Divides CPU bandwidth between NRT flows based on their respective weights.
- Non-real-time flows assigned deadlines by WFQ and then scheduled by EDF along with RT Flows.

Task Scheduler Based on a Hierarchy of WFQ and EDF



Task Scheduling in Network Processors

Recall that α^u is the upper arrival curve, $d(f_i)$ is the deadline, Δ is the time interval, and $w(f_i)$ is the cost function.

$$\overline{\alpha}_i^u(\Delta) = \begin{cases} 0 & \text{if } \Delta \le d(f_i) \\ w(f_i)\alpha_i^u(\Delta - d(f_i)) & \text{otherwise} \end{cases}$$

$\alpha_{RT}(\Delta)$ = sum of all alpha bars
$\beta^l(\Delta)$ has to be greater than α_{RT}

Task Scheduling in Network Processors

- Recall NRT arrival curves are not specified so they can be specified by the following function.

$$\alpha_{NRT}(\Delta) = \min_{t \geq \Delta}\{\beta^{l}(t) - \alpha_{RT}(t)\}$$

Task Scheduling in Network Processors

- For each packet selected by the WFQ scheduler for processing, if the packet belongs to flow f_i and has a packet processing requirement of $w(f_i)$ then it is assigned a deadline:

$$d(f_i) = \min\{\Delta : \alpha_{NRT}(\Delta) \geq w(f_i)\}$$

Task Scheduling in Network Processors

- Proposition 2 – Schedulability
  - If the set of real-time flows is preemptively schedulable then the algorithm also schedules the real time flows such that all deadlines are met.

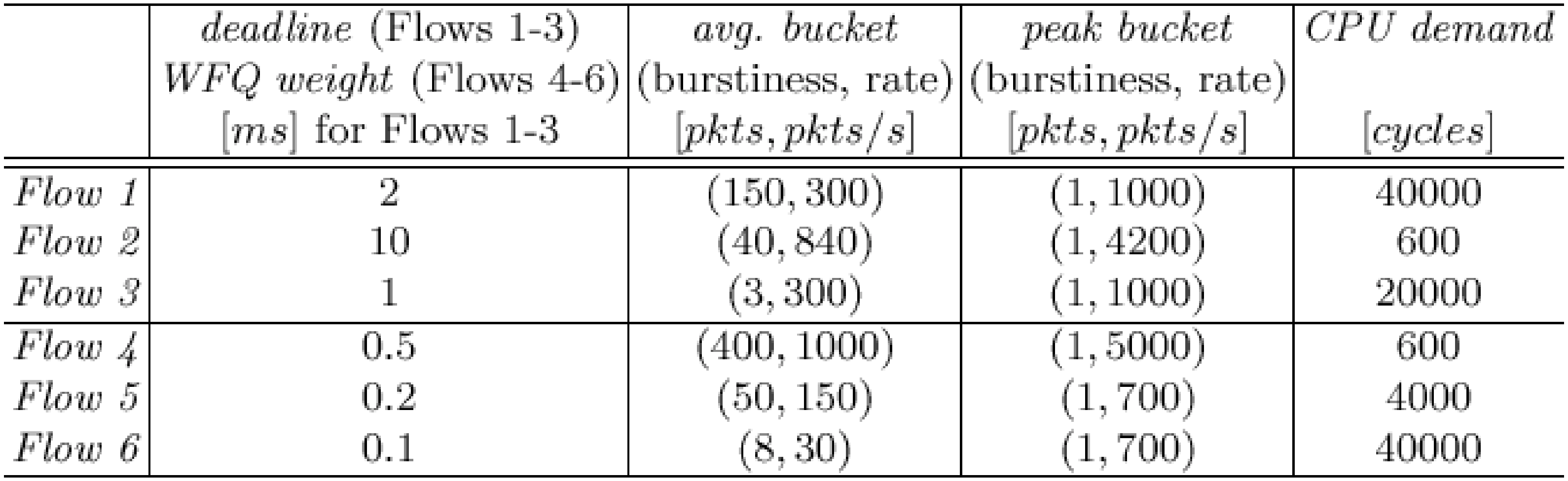
Task Scheduling in Network Processors

- This scheduling algorithm is preemptive.
- Arbitrary preemptions might be costly for any practical implementation.
- Given that the execution time of a node is small compared to the total execution time of the whole task graph, the previous analysis gives a good approximation of an algorithm where preemption is allowed only at the end of each node.

Experimental Evaluation

- Evaluated using the Moses tool-suite (modeling and simulation of discrete event systems)
- Experimental setup consists of six flows
  - 3 Real-time
  - 3 Non-real-time
- Each flow specified by a TSpec with parameters in terms of packets rather than bytes
- A Tspec is described by a conjunction of two token buckets and an incoming packet complies with the specified profile only if there are enough tokens in both buckets.

Specifications of the real-time and non-real time flows



	deadline (Flows 1-3) *WFQ weight* (Flows 4-6) [*ms*] for Flows 1-3	*avg. bucket* (burstiness, rate) [*pkts*, *pkts*/*s*]	*peak bucket* (burstiness, rate) [*pkts*, *pkts*/*s*]	*CPU demand* [*cycles*]
Flow 1	2	$(150, 300)$	$(1, 1000)$	40000
Flow 2	10	$(40, 840)$	$(1, 4200)$	600
Flow 3	1	$(3, 300)$	$(1, 1000)$	20000
Flow 4	0.5	$(400, 1000)$	$(1, 5000)$	600
Flow 5	0.2	$(50, 150)$	$(1, 700)$	4000
Flow 6	0.1	$(8, 30)$	$(1, 700)$	40000

Flow 1-3 Real Time
Flow 4-5 Non-real-time

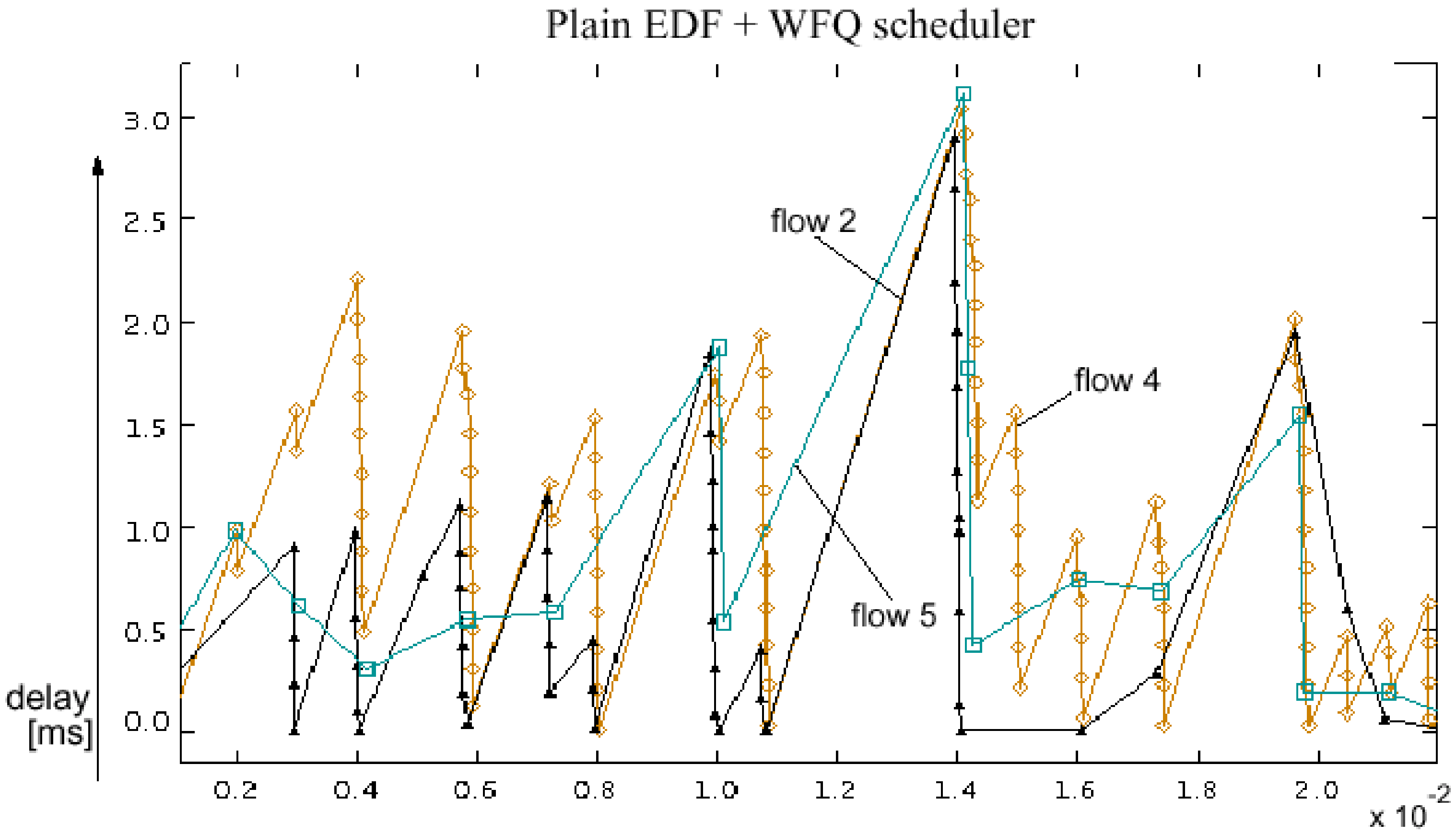
1- Encryption
2- Video Traffic
3- Voice Encoding

4- FTP
5- HTTP
6-Email Traffic

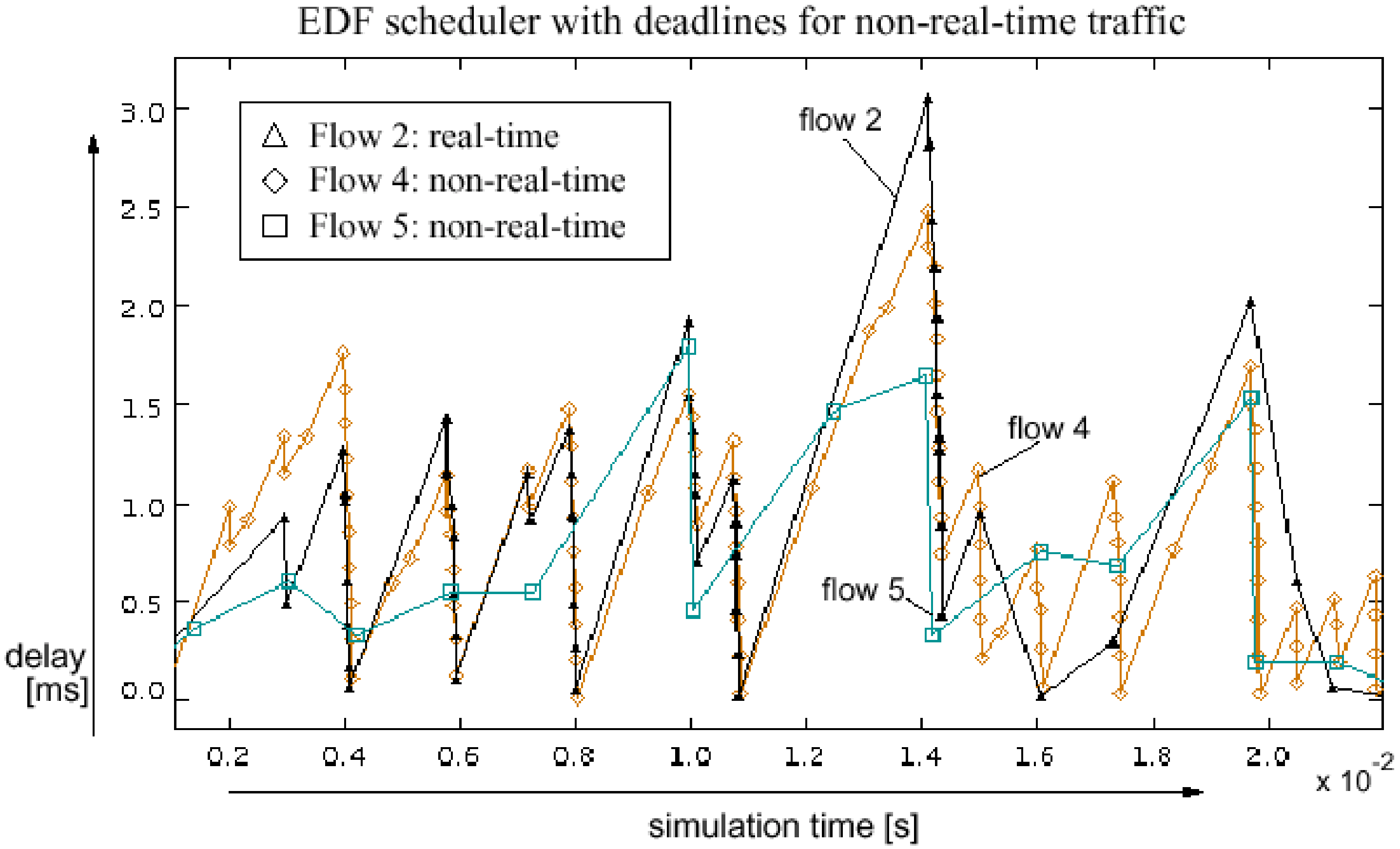
Experimental Evaluation

- Compared our algorithm with a plain EDF for real time and WFQ for non-real-time (i.e. not hierarchical).
- Horizontal axis shows the simulation time
- Vertical axis represents the delay experienced by the packet getting processed.

Experimental Evaluation



Experimental Evaluation



Experimental Evaluation

- Bottom line:
  - Non-real-time flows happen faster at the expense of real time flows, but not in such a way that deadlines are violated.
  - Flows 4 & 5 (NRT) have shorter response time in the hierarchical algorithm at the expense of higher delay times for flow 2.

Design Space Exploration

- It is expected that the next generation of network processors will consist of general purpose processing units and dedicated modules for executing run-time extensive functions.
- Therefore we need to select appropriate functional units such that the performance is maximized under various constraints (cost, delay, power, etc)
- How do we explore the design space of NP?

Design Space Exploration

- Concentrate on the following questions:
  - How can we estimate the performance of a network processor?
  - How can we estimate delay and memory consumption of a hardware/software architecture?
- Adopt a model based approach in combination with concepts of multi-objective optimization.

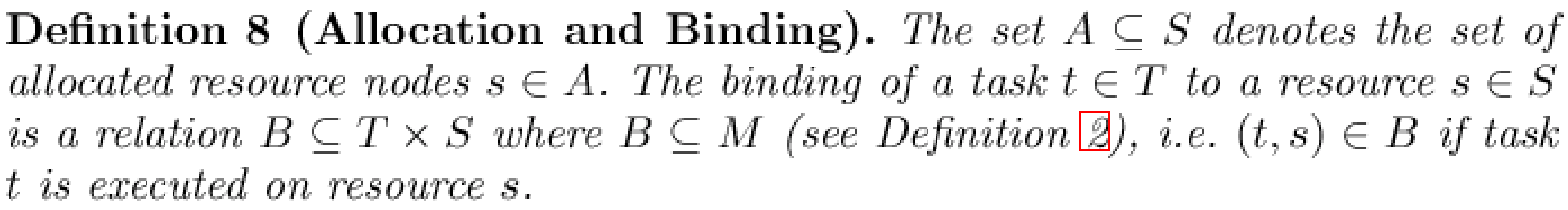
Design Space Exploration

- Think of design space exploration as:
  - Allocate resource nodes $s \in S$ and bind the tasks $t \in T$ of the flows $f \in F$ to the allocated resource nodes such that upper and lower arrival curves for NRT flows are maximized, cost, memory, and power consumption are minimized and the deadlines $d(f)$ associated to the flows are satisfied.
- Note the RT flows often have a fixed arrival rate.

Design Space Exploration

- NP consist of heterogeneous elements (RISC cores, DSPs, dedicated units, etc).
- The purpose of the allocation is to select the right subset of these modules.

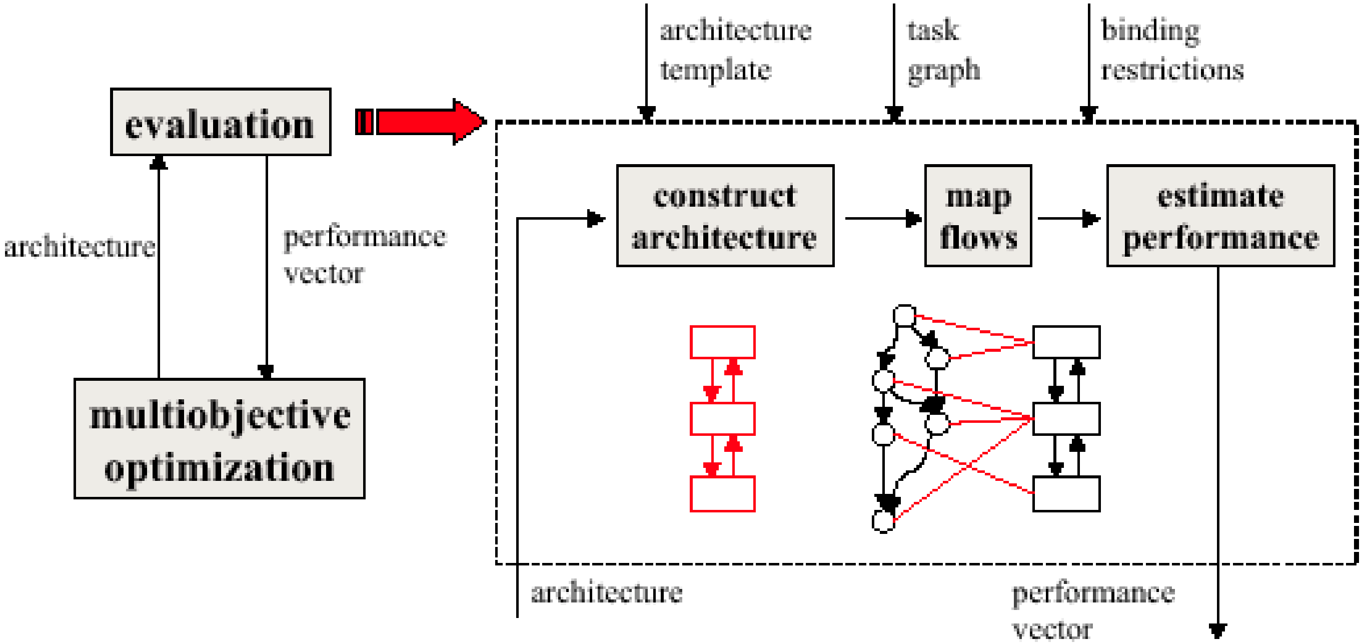
Design Space Exploration



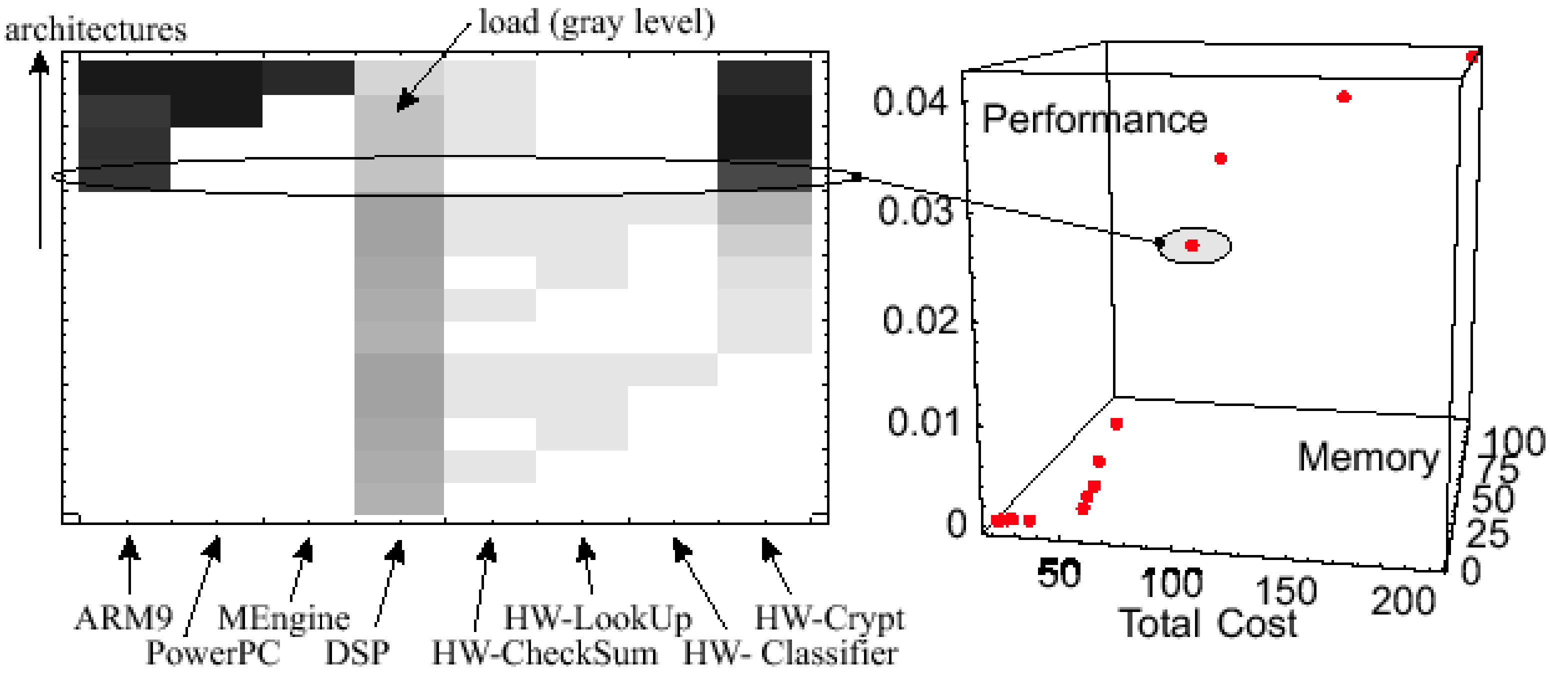
Definition 8 (Allocation and Binding). *The set* $A \subseteq S$ *denotes the set of allocated resource nodes* $s \in A$*. The binding of a task* $t \in T$ *to a resource* $s \in S$ *is a relation* $B \subseteq T \times S$ *where* $B \subseteq M$ *(see Definition 2), i.e.* $(t, s) \in B$ *if task* t *is executed on resource* s*.*

This allows you to formalize the design space exploration problem so that branch and bound search algorithms can be used to form the points on Pareto curves.

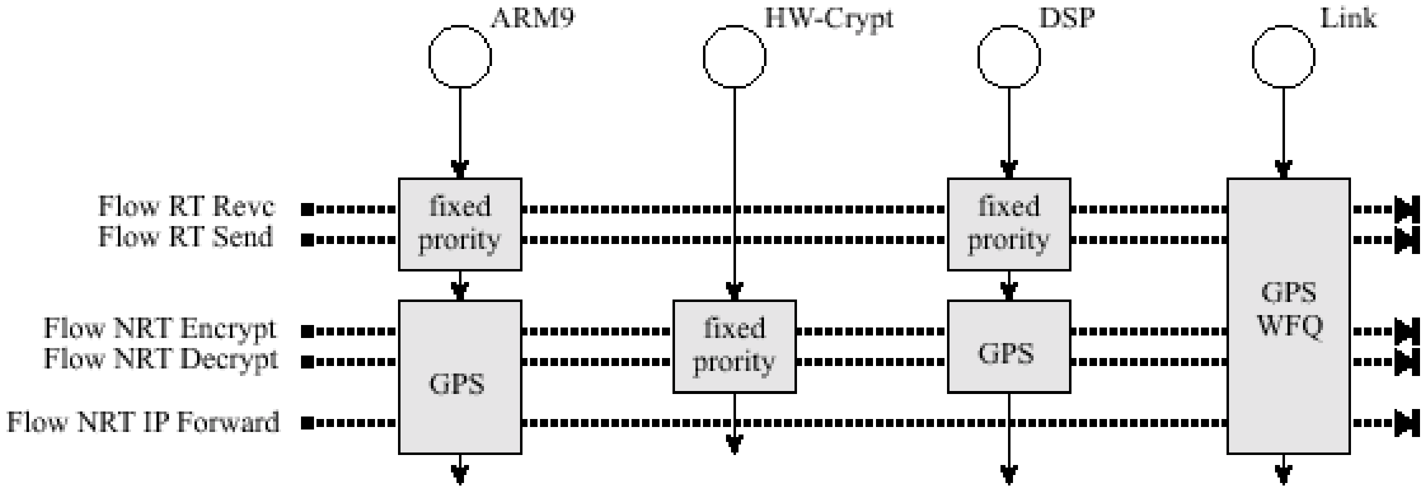
Design Space Exploration



Design Space Exploration



Simple Processing Network



Conclusion

- Introduced a packet flow model.
- Scheduling of real time and non-real time flows.
- Design Space exploration methodology.
- Many open issues in network processing!

- Questions??