Stochastic Models for Optimization of Software-defined Radio Operation

Yanzhou Liu[1]

With contributions from:
Marilyn Wolf[3], Shuvra Bhattacharyya[1,2] Adrian Sapio[1], Lin Li[1]

1. University of Maryland, USA
2. Tampere University of Technology, Finland
3. Georgia Institute of Technology, USA

Supported by NSF grants CNS151304 and CNS1514425

WinnComm 2017, 11/15/2017

A. JAMES CLARK SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING
UMIACS University of Maryland Institute for Advanced Computer Studies

Outline

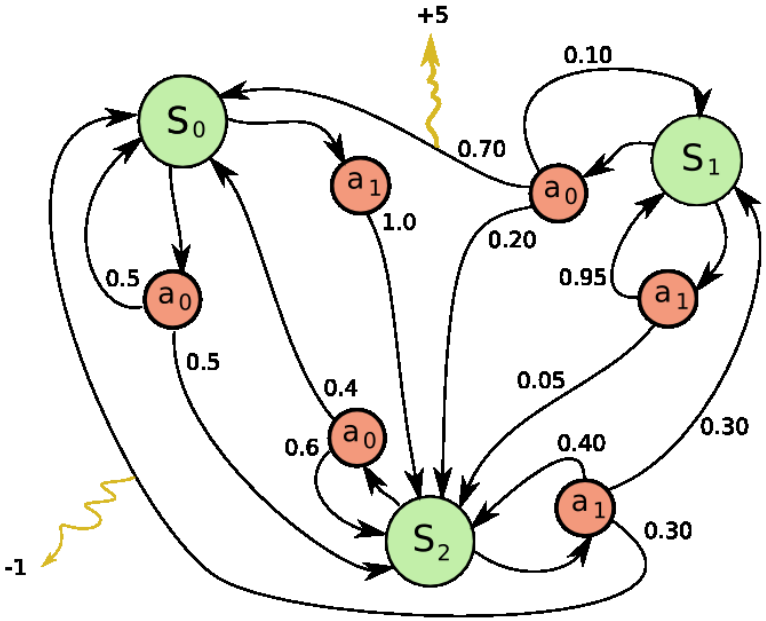
- Introduction
  - Motivation and Contribution
- Background
- Framework for Adaptive Signal Processing Systems
- Case Study and Experimental Results
- Conclusion

Motivation

- Runtime adaptation of computing systems to their dynamic environment.
  - Deploy intelligent control logic alongside signal processing functionality.
  - Defer some optimization and design decisions to runtime.
- Challenges for system operation in the communication system.
  - Operation should be optimized for both communication parameters as well as operational parameters.
- Stochastic models can be used to capture uncertainty and variation in communication and computation systems and communication channels.

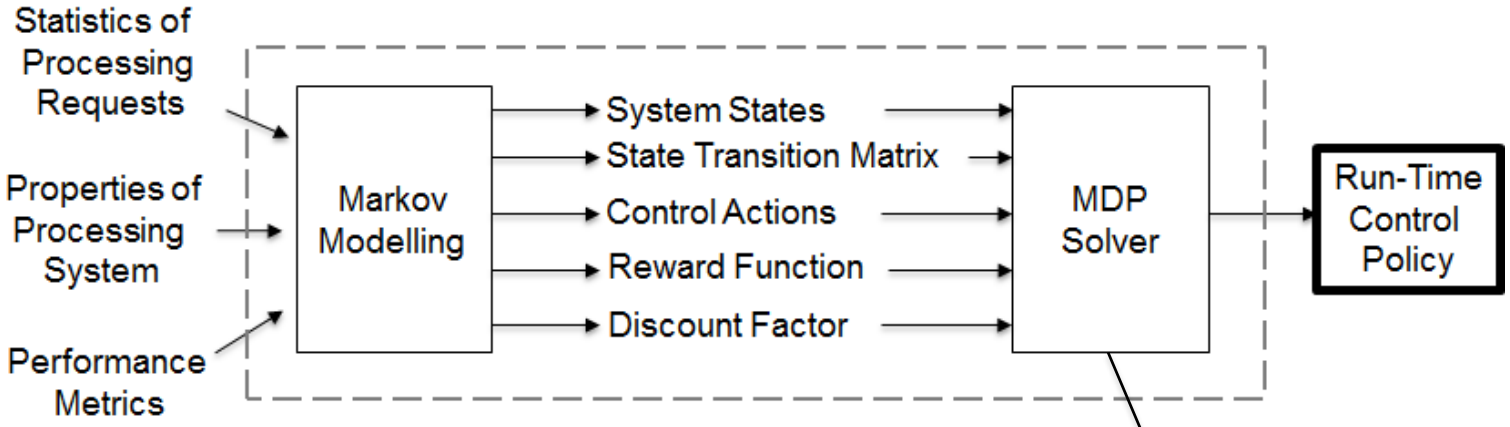
Background: Markov Decision Process

- MDPs provide a framework for decision making under uncertainty.
- Extremely general and flexible.
  - Hybrid deterministic and Stochastic systems.
- Probabilistic transitions combined with inputs.
  - Given an input at a state, next state is chosen probabilistically.
  - Associated with reward function.
- By defining a system using the constructs of an MDP, a control policy can be generated to maximize the reward using well-known, efficient solver algorithms.
  - Linear Programming or Dynamic Programming.
  - Multitude of open source toolboxes with solvers (C/C++, Python, MATLAB, …).



Markov Modeling for Embedded Signal Processing

- Algorithmic method to convert measurable quantities into control policies.
- Enables autonomous adaptation of system level configurations.



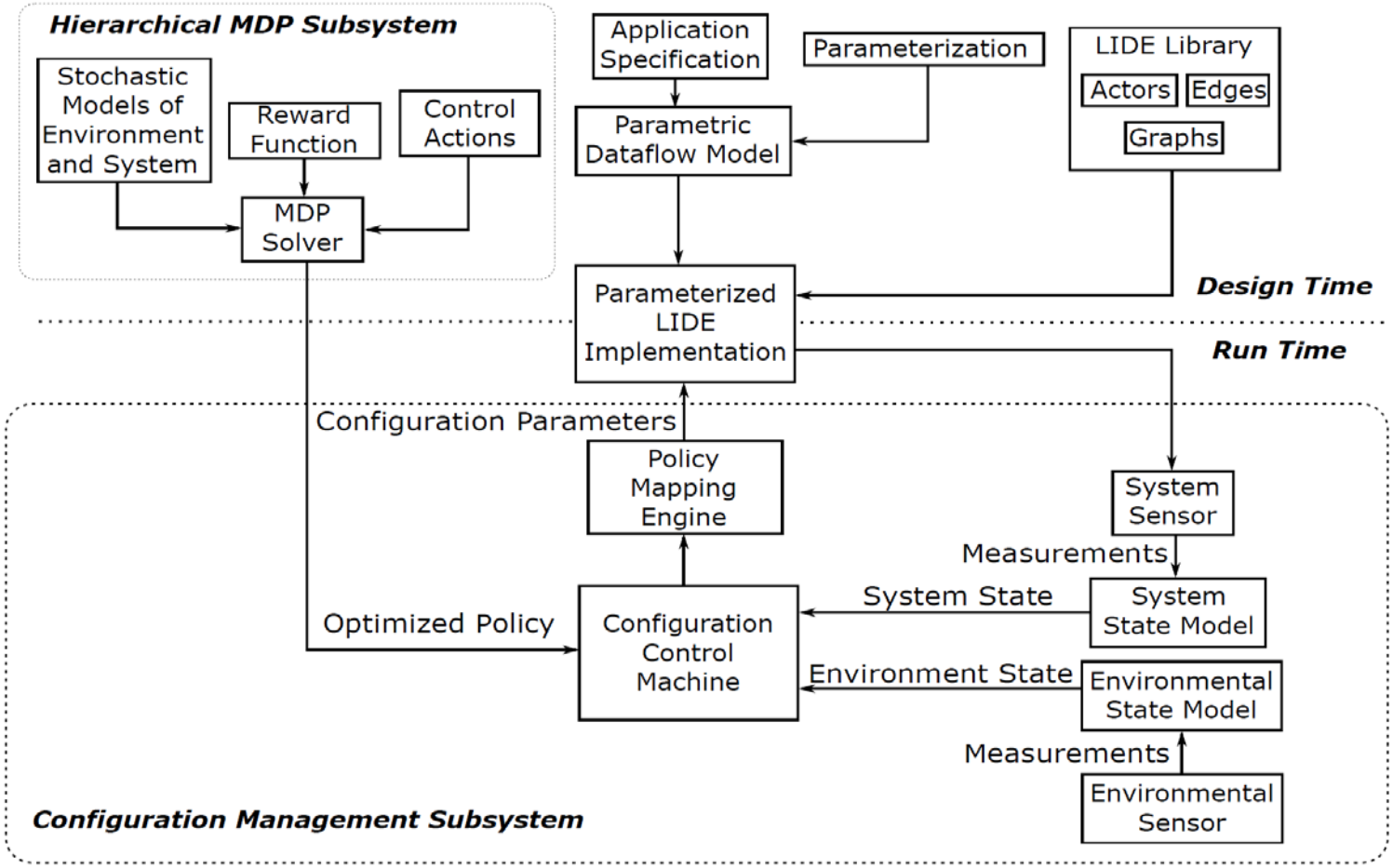
- Processing Request can be implicit or explicit.
- Properties can be physical constraints.
  - Programmable logic size.
  - Transition time between states.

$$\pi(s) := \arg\max_a \left\{ \sum_{s'} P_a(s,s') \left(R_a(s,s') + \gamma V(s') \right) \right\}$$

$$V(s) := \sum_{s'} P_{\pi(s)}(s,s') \left(R_{\pi(s)}(s,s') + \gamma V(s') \right)$$

Framework for Adaptive Signal Processing Systems

Hierarchical MDP framework for Compact System-level Modeling (HMCSM).



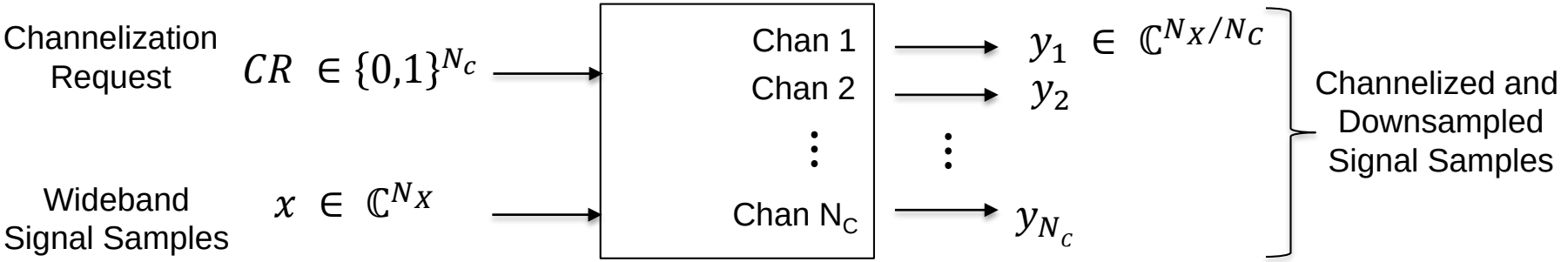
Policy Generation in HMCSM

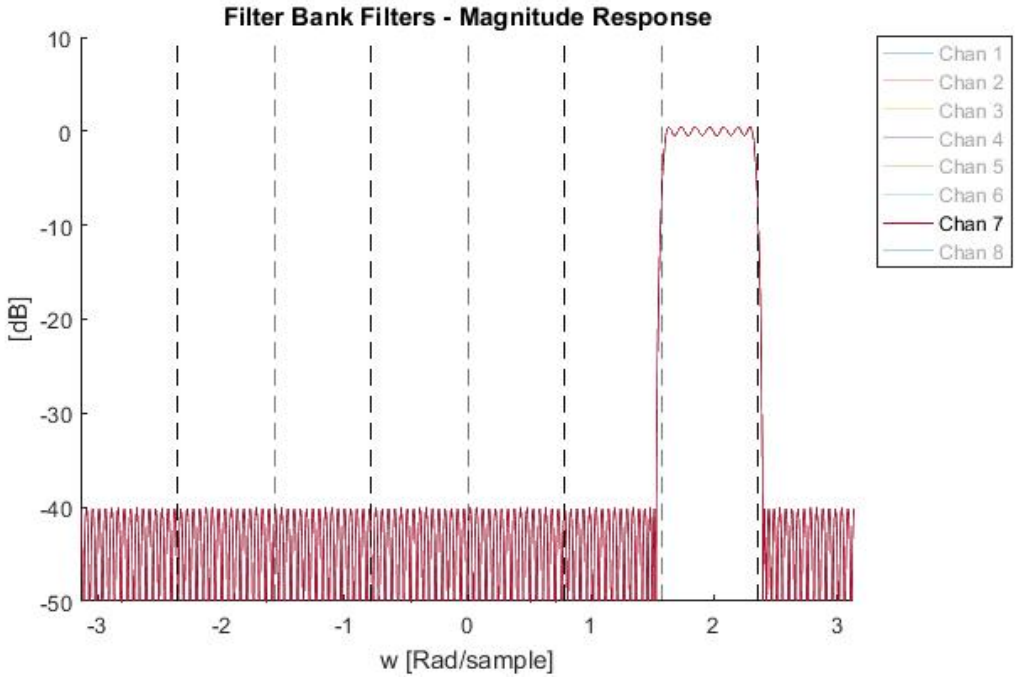
- Derived by an MDP solver, which is a software module that automatically generates optimized policies from MDP model specifications.
- Inputs to the MDP solver in HMCSM:
  - Stochastic models of environment and system
  - Reward function
  - Control actions
- The stochastic models of the environment and system include, for each of the two models, the definition of the state space and state transition matrix.
- The system model can be updated at run-time.
  - Dynamic reconfiguration of the model allows robust optimization of system operations.

Factored MDP Models

- A single Markov decision process (MDP) is transformed into nested multiple MDPs that can be independently solved.
- Results in smaller storage requirements and faster solver runtime.
  - Demonstrated in our experiments using HMCSM.
- Enables more efficient dynamic updating of models.

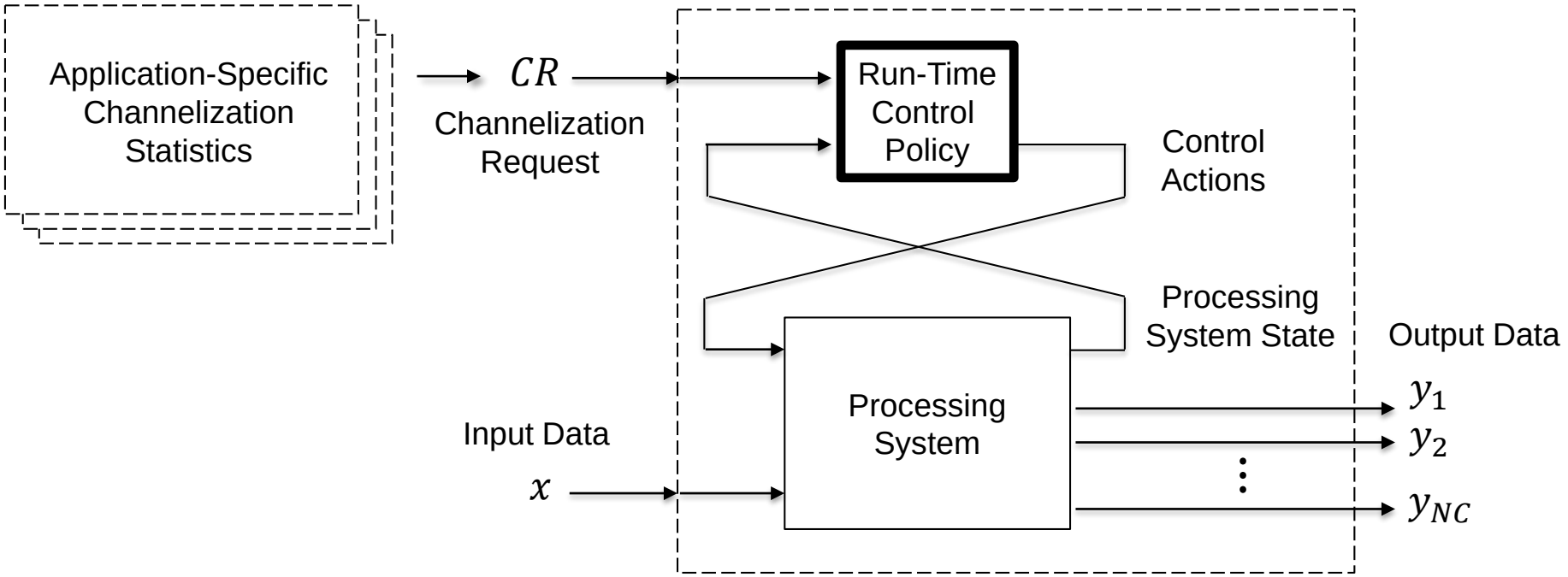
Case Study: Digital Channelizer Block Diagram



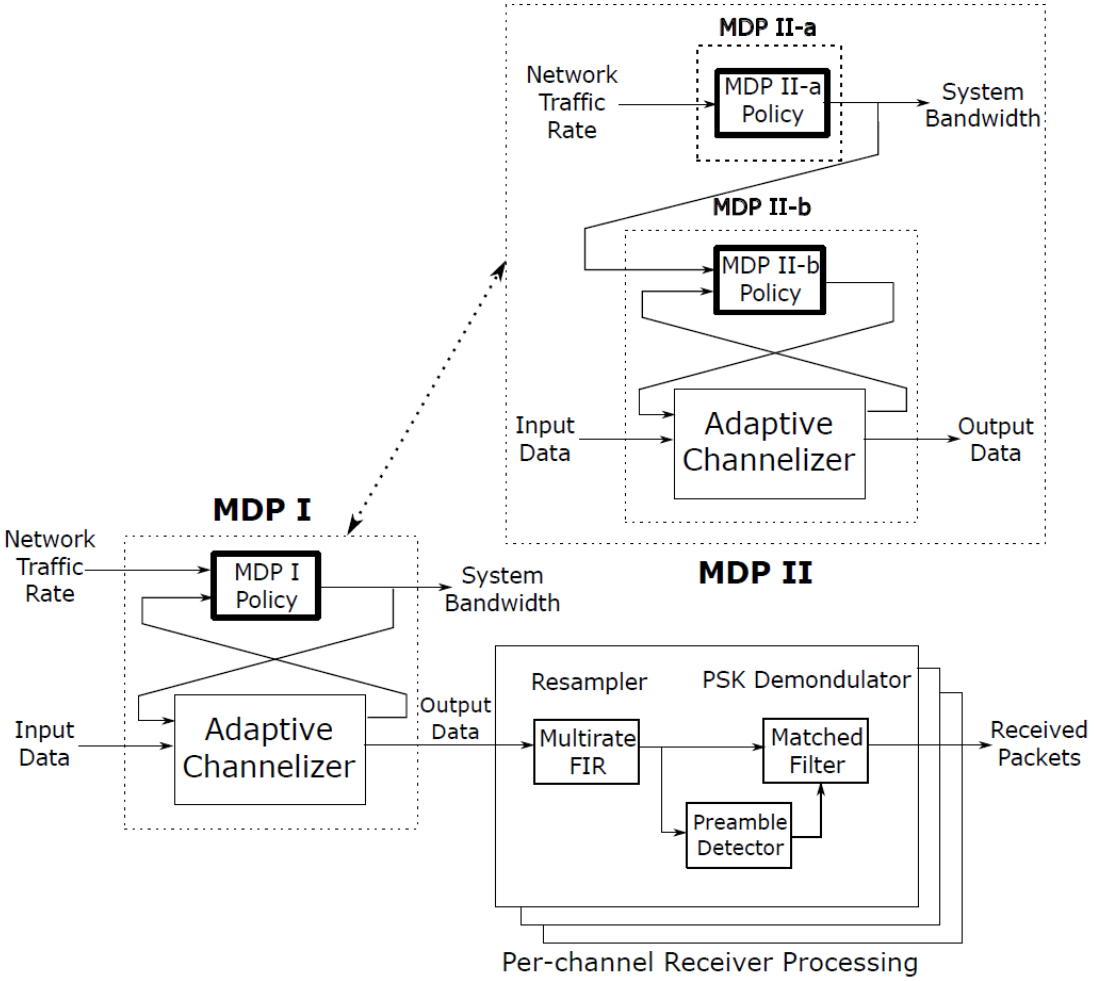


MDP Model for Digital Channelizer

- A simulation for a dynamic channelizer was created with 3 top-level processing states
  - FIR Downconverter (with 8 sub-configurations)
  - DFT Filter Bank
  - Sleep
- 2-frame delay was assumed to switch between algorithms
- Power measurements taken by running C language implementations of the algorithms on ARM Cortex-M3



Factored MDP Model for Digital Channelizer



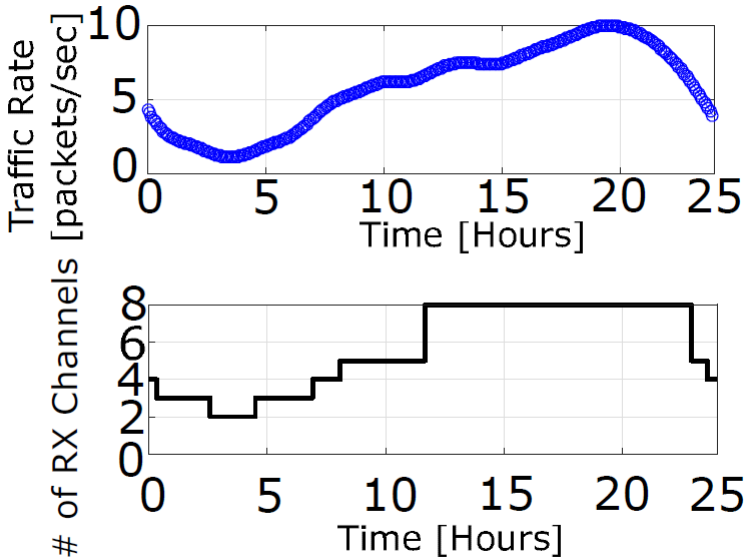
Experiments and Results—Average Power with Different Configurations

Processing Configuration	Number of Channels Processed	Average Power
DCM	1	1.4406 W
DCM	2	1.4781 W
DCM	3	1.5203 W
DCM	4	1.5660 W
DCM	5	1.6025 W
DCM	6	1.6524 W
DCM	7	1.7013 W
DCM	8	1.7453 W
DFTFB	8	1.6754 W

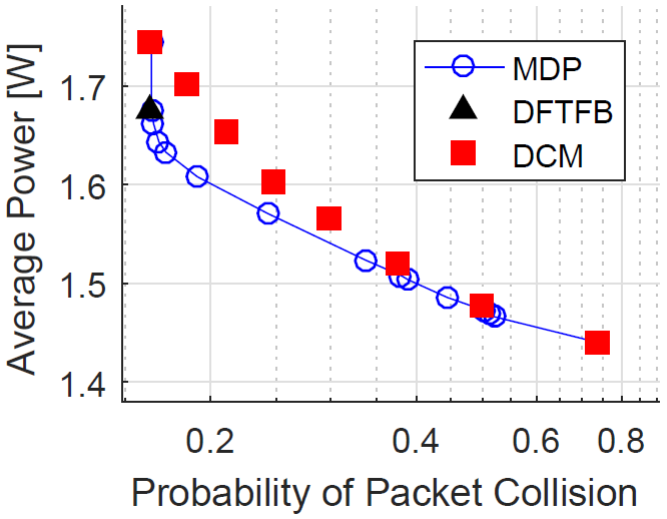
Average processing power of all the available configurations.

- Platform for Channelizer: Raspberry Pi 3 Model B
- Device for Power Measurement: Tektronix Keithley Series 2280 Precision Measurement DC Power Supply

Experiments and Results—Simulation Results of MDP Solutions



Simulation results for MDP-I.



Comparison among MDP-generated policies and fixed-configuration designs.

In addition to providing better energy efficiency compared to the fixed configuration designs, our MDP approach:

- can be configured systematically to generate a much larger set of trade-off options (Pareto-optimized fronts);
- ensures optimality (with respect to the given reward function).

- **Solver running time**
  - MDP Solver: MATLAB-based open source solver, named MDPSOLVE
  - Platform for MDP Solver:
    - Processor: i7-4710HQ running at 2.50GHz
    - RAM: 12.0 GB
    - OS: 64-bit Windows 10
  - MDP-I solver running time: 294ms
  - MDP-II-a solver running time: 50.8ms
  - MDP-II-b solver running time: 41.5ms
  - In a deployment with a fixed processing system (MDP-II-b) and changing external environment (MDP-II-a), the hierarchical MDP scheme reduces the solver time from 294ms to 50.8ms, which is a factor of over 5.7X smaller.
- **Model size**
  - MDP-I: 1.63MB
  - MDP-II: 265kB
  - MDP-II reduces model size by a factor of over 6.1X.

Conclusion

- Hierarchical Markov decision process (MDP) framework for design and implementation of adaptive embedded signal processing systems
- Enables robust optimization across a wider design space compared to conventional methods
- Use of factored MDPs leads to more compact memory requirements, faster solver time, and more efficient dynamic updating of the models.
- Prototyped by building on the Lightweight Dataflow Environment (LIDE) for model-based signal processing system design

References

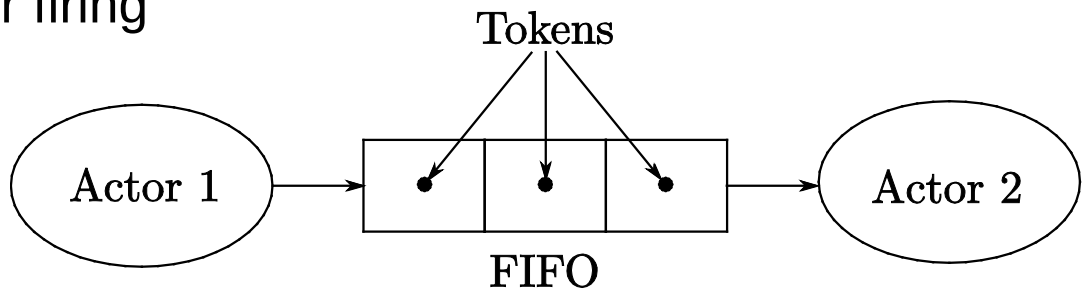
- [1]. A. Sapio, M. Wolf, and S. S. Bhattacharyya. Compact modeling and management of reconfiguration in digital channelizer implementation. In *Proceedings of the IEEE Global Conference on Signal and Information Processing*, pages 595-599, Washington, D.C., December 2016.
- [2]. L. Li, A. Sapio, J. Wu, Y. Liu, K. Lee, M. Wolf, and S. S. Bhattacharyya. Design and implementation of adaptive signal processing systems using Markov decision processes. In *Proceedings of the International Conference on Application Specific Systems, Architectures, and Processors*, pages 170-175, Seattle, Washington, July 2017.

Thanks!

Q&A

Background: Dataflow Modeling

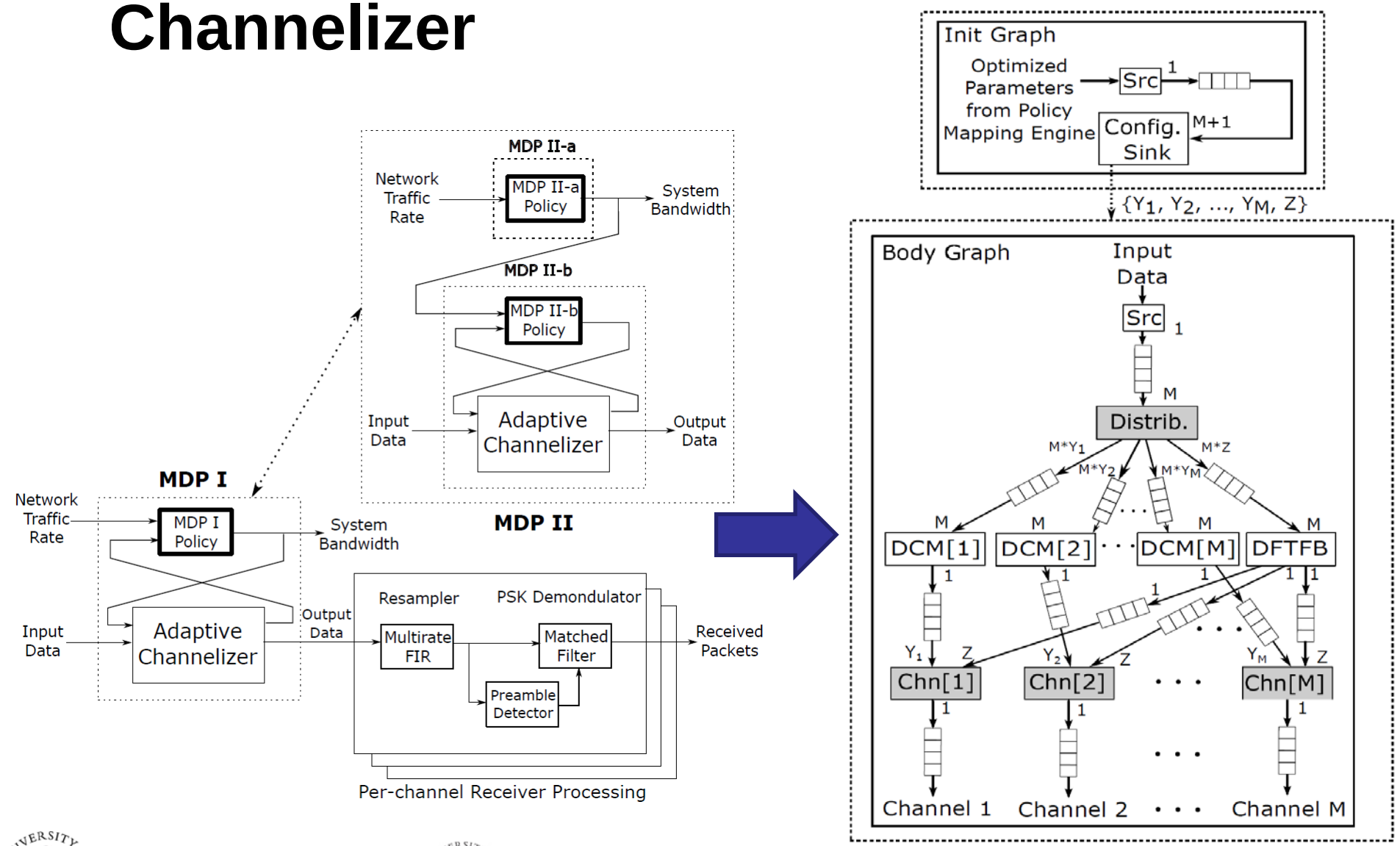
- Basic concepts of dataflow modeling:
  - Digital Signal Processing (DSP) system ⟷ directed dataflow graph
  - Computational functions ⟷ nodes (actors)
  - Communication channels between actors ⟷ edges (FIFOs)
  - **Actor Firing:** Actor execution as a discrete unit of computation
  - **Token:** The encapsulation of some well-defined amount of data
    Consumption/Production Rate: Number of tokens consumed/produced from/to the input/output FIFO during a single actor firing



Background
LightWeight DataFlow (LWDF)

- **LightWeight DataFlow (LWDF):** a programming methodology for integration of, experimentation with, and optimization of dataflow modeling approaches.
- **Actor Mode:** Determines the dataflow behavior of the actor.
- **Enable function:** Checks actor firing condition according to its current mode. This function can be bypassed at run time if static scheduling analysis can ensure the result.
- **Invoke function:** Executes an actor firing according to its current mode.
- **Lightweight Dataflow Environment (LIDE):**
  - Provides a compact set of LWDF application programming interfaces (APIs) that is used for constructing, connecting, and executing dataflow components such as actors, edges, and graphs.
  - LIDE APIs have been implemented in a variety of implementation languages, including C, Verilog, and CUDA.

Factored MDP Model for Digital Channelizer



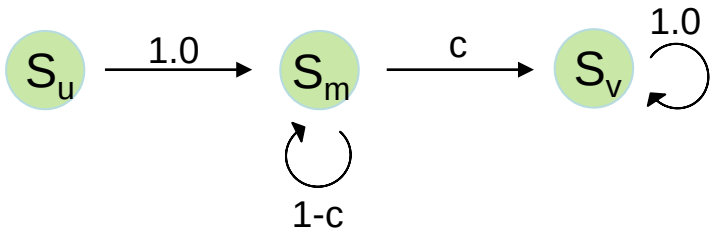
Transition States

- Added to model reconfigurations within a processing resource with duration longer than one frame.
- Represents transition through undesirable temporary state to more desirable final state.

(Conditional) State Transition Matrix

Current State \ Next State	S_u				S_m	S_v		
S_u	0	0	0	0	1	0	0	0
S_m	0	0	0	0	1-c	c	0	0
S_v	0	0	0	0	0	1	0	0

(Conditional) Markov Chain Diagram



$$c = \left[floor\left(\frac{T_{u,v|w}}{T_F} \right) \right]^{-1}$$

Rewards: Multiobjective Optimization

- In an MDP framework, a Reward function is a mapping:

$$R(s,a,s'): \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$$

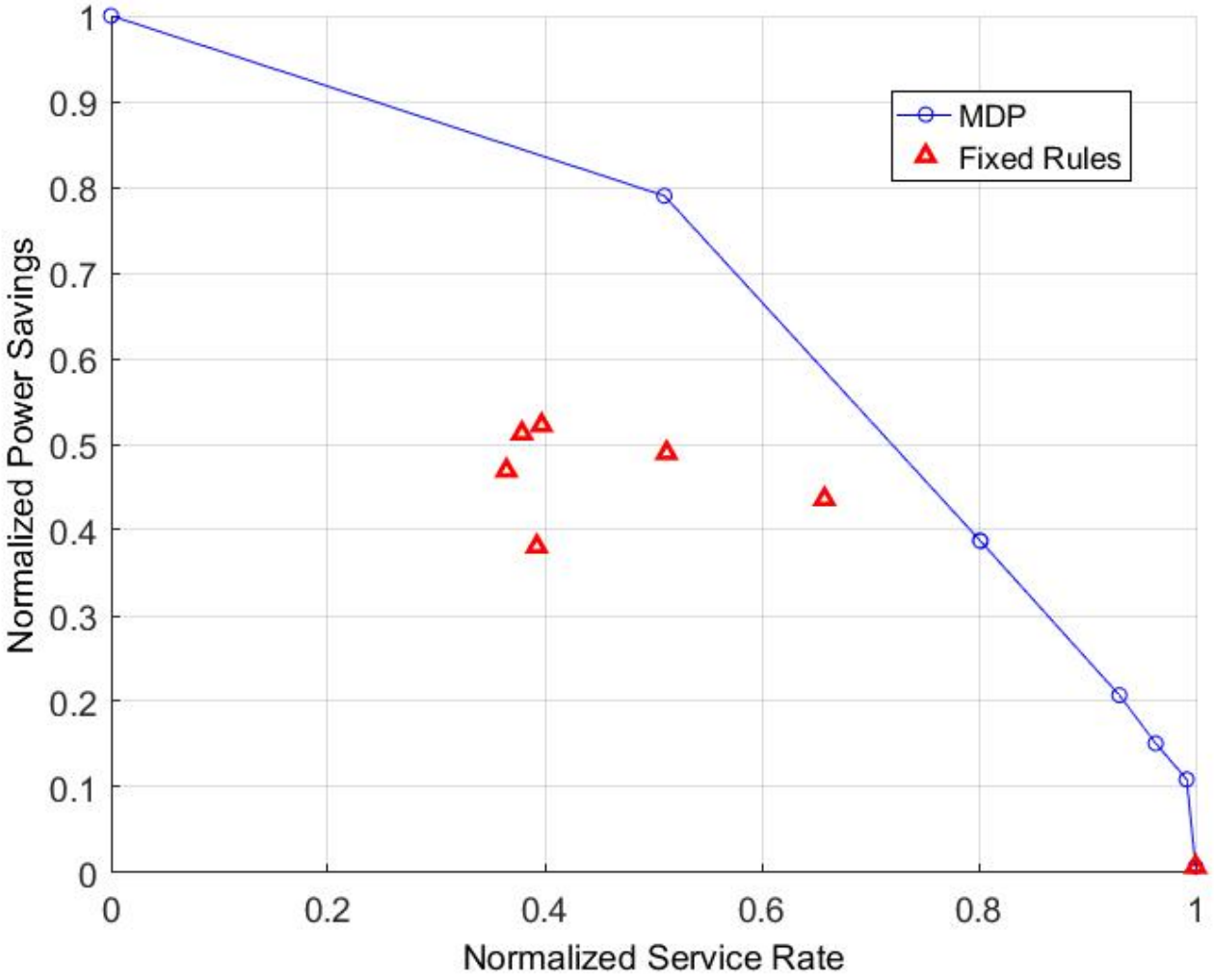
- We use a *scalarization* approach to steer the MDP solver with multiple performance metrics:
  - i=1: “Productivity” = Number of output channels produced
  - i=2: “Power Consumption” = Average power consumed by the processing platform

$$f_i(s,a,s'): \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$$

$$R(s,a,s') = \sum_i r_i \, f_i(s,a,s')$$

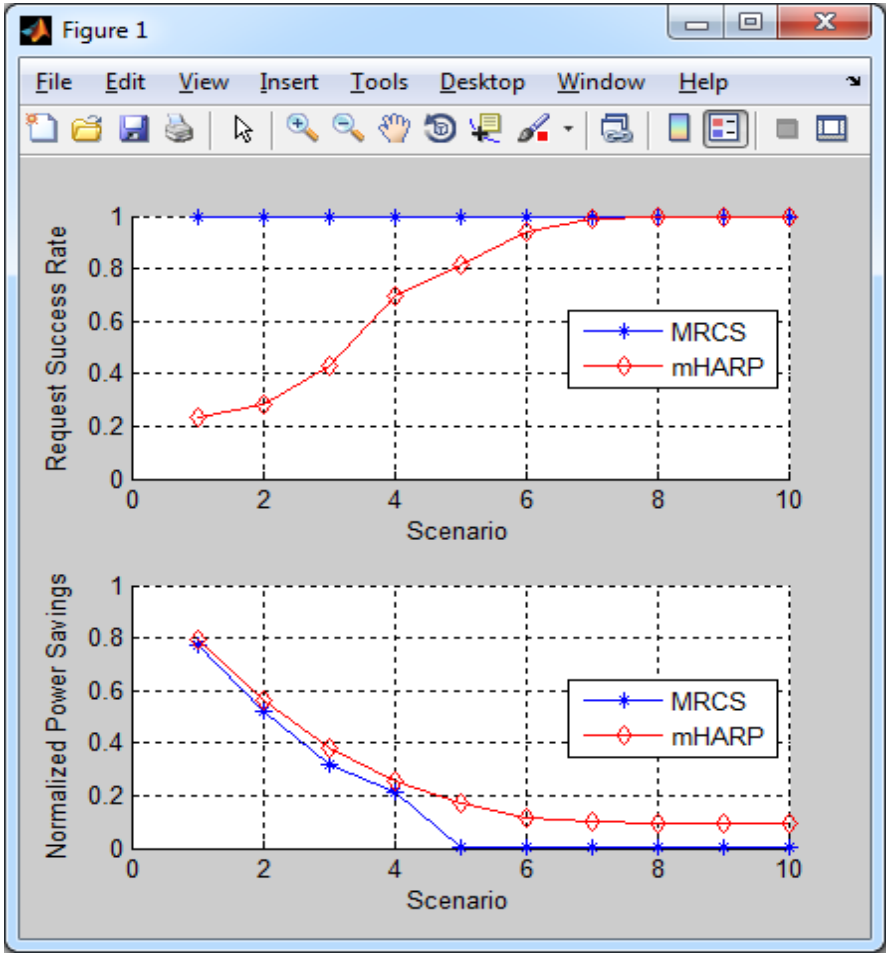
- Metrics must adhere to convention of 1=most rewarded, 0=least rewarded
- Metrics can be known at design time or measured at runtime
- At design time, the application engineer instructs the system in terms of $\underline{r}$ = *Relative importance of each metric*, instead of static rules for reconfiguration.

Results 1: MDP vs. Fixed Rules



Results 2: MDP vs. mHARP [Hsieh 2014]

Application 1:
Dynamic Spectrum Access



Application 2:
Sequential Sensing