

# CONTENTS

---

<b>Preface</b>	<b>xi</b>
<b>Acknowledgments</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Sensor networks and traditional distributed systems	2
1.2 Programming of distributed sensor networks	7
1.2.1 Layers of programming abstraction	7
1.2.1.1 Service-oriented specification	7
1.2.1.2 Macroprogramming	8
1.2.1.3 Node-centric programming	10
1.2.2 Lessons from parallel and distributed computing	12
1.3 Macroprogramming: What and why?	14
1.4 Contributions and outline	16

<b>2</b>	<b>The Abstract Task Graph</b>	<b>21</b>
2.1	Target applications and architectures	21
2.2	Key concepts	23
2.2.1	Data-driven computing	23
2.2.1.1	Program flow mechanisms	23
2.2.1.2	Why data-driven?	26
2.2.2	Mixed imperative–declarative specification	28
2.3	Syntax	29
2.3.1	The Structure of an ATaG program	29
2.3.2	More on task annotations	34
2.3.3	Illustrative examples	39
2.4	Semantics	45
2.4.1	Terminology	45
2.4.2	Firing rules	46
2.4.3	Task graph execution	48
2.4.4	get () and put ()	48
2.5	Programming idioms	49
2.5.1	Object tracking	51
2.5.2	Interaction within local neighborhoods	52
2.5.3	In-network aggregation	52
2.5.4	Hierarchical data fusion	55
2.5.5	Event-triggered behavior instantiation	56
2.6	Future work	59
2.6.1	State-based dynamic behaviors	59
2.6.2	Resource management in the runtime system	61
2.6.3	Utility-based negotiation for task scheduling and resource allocation	63
<b>3</b>	<b>DART: The Data-Driven ATaG Runtime</b>	<b>65</b>
3.1	Design objectives	65
3.1.1	Support for ATaG semantics	65
3.1.2	Platform independence	66
3.1.3	Component-based design	67
3.1.4	Ease of software synthesis	68
3.2	Overview	69
3.3	Components and functionalities	72

3.3.1	Task, data, and channel declarations	72
3.3.2	UserTask	75
3.3.2.1	Service	75
3.3.2.2	Interactions	75
3.3.2.3	Implementation	77
3.3.3	DataPool	79
3.3.3.1	Service	79
3.3.3.2	Interactions	79
3.3.3.3	Implementation	79
3.3.4	AtagManager	82
3.3.4.1	Service	82
3.3.4.2	Interactions	82
3.3.4.3	Implementation	83
3.3.5	NetworkStack	87
3.3.5.1	Service	87
3.3.5.2	Interactions	87
3.3.5.3	Implementation	87
3.3.6	NetworkArchitecture	88
3.3.6.1	Service	88
3.3.6.2	Interactions	88
3.3.6.3	Implementation	89
3.3.7	Dispatcher	90
3.3.7.1	Service	90
3.3.7.2	Interactions	91
3.3.7.3	Implementation	91
3.4	Control flow	93
3.4.1	Startup	94
3.4.2	get () and put ()	97
3.4.3	Illustrative example	100
3.5	Future work	101
3.5.1	Lazy compilation of channel annotations	101
3.5.2	Automatic priority assignment for task scheduling	102

<b>4</b>	<b>Programming and Software Synthesis</b>	<b>105</b>
4.1	Terminology	106
4.2	Meta-modeling for the ATaG domain	106
4.2.1	Objectives	106
4.2.2	Application model	108
4.2.3	Network model	110
4.3	The programming interface	112
4.4	Compilation and software synthesis	115
4.4.1	Translating task annotations	117
4.4.2	Automatic software synthesis	117
4.4.3	The ATaG simulator	121
4.4.4	Initialization	122
4.4.4.1	Situatedness	123
4.4.4.2	Network interface	124
4.4.4.3	Network architecture	124
4.4.4.4	Sensor interface	124
4.4.5	Visualizing synthesized application behavior	128
<b>5</b>	<b>Case Study: Application Development with ATaG</b>	<b>135</b>
5.1	Overview of the use case	136
5.2	Designing the macroprograms	136
5.2.1	Temperature gradient monitoring	136
5.2.2	Object detection and tracking	139
5.3	Specifying the declarative portion	142
5.4	Imperative portion: Temperature gradient monitoring	143
5.4.1	Abstract data items: Temperature and fire	143
5.4.2	Abstract task: Monitor	144
5.4.3	Abstract task: Temperature sampler	151
5.4.4	Abstract task: Alarm actuator	153
5.5	Imperative portion: Object detection and tracking	155
5.5.1	Abstract data items: TargetAlert and TargetInfo	155
5.5.2	Abstract Task: SampleAndThreshold	157
5.5.3	Abstract Task: Leader	157
5.5.4	Abstract Task: Supervisor	165
5.6	Application Composition	165
5.7	Software Synthesis	171

<b>6 Concluding Remarks</b>	<b>175</b>
6.1 A framework for domain-specific application development	176
6.2 A framework for compilation and software synthesis	177
<b>References</b>	<b>179</b>
<b>Index</b>	<b>185</b>