WSAN向けマクロプログラミング言語の提案

後藤司・森口草介・渡部卓雄（東京工業大学）

概要

無線センサーアクターネットワーク(WSAN)は物理環境に配備された複数個の計算機が相互に通信することで、物理環境の観測や環境への働きかけを行う分散システムである。その実現にはセンサーノードに加え、環境への働きかけなどを行うアクターノードを含む複雑なノード間協調が必要である。アクターノードを含まない無線センサーネットワーク(WSN)については、マクロプログラミングと呼ばれる、WSN全体を一つの計算システムとみなしてその動作を記述する手法がある。本研究ではWSAN向けのマクロプログラミングのための言語を提案する。

WSAN

物理環境の観測や環境への働きかけを行う分散システム

- センサーノード：物理環境の計測、データの送信
- アクターノード：計測値の集約、物理装置の作動

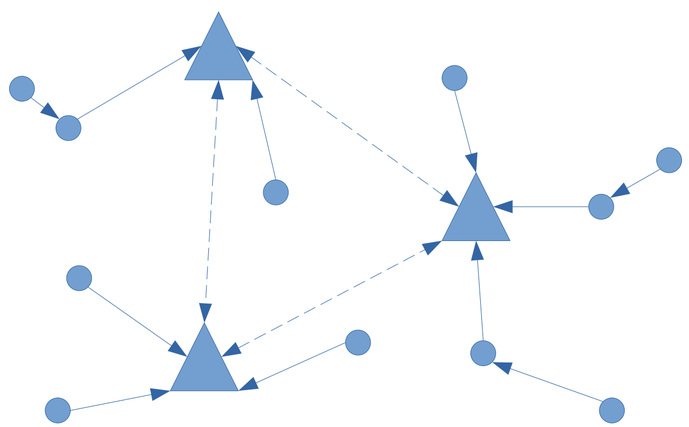
アクターノードはより豊富な資源を持ち、センサーノードより複雑な処理を行う

センサー・アクター間協調、アクター・アクター間協調が必要であり、その記述は複雑になる

例：温室管理システム

センサーが土壌水分量や気温を計測してアクターに送信

アクターはヒーターやスプリンクラーなどを制御



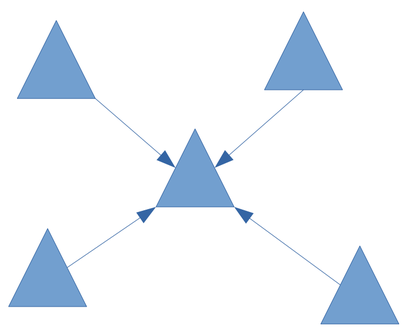
WSAN向けマクロプログラミング言語：Brigade

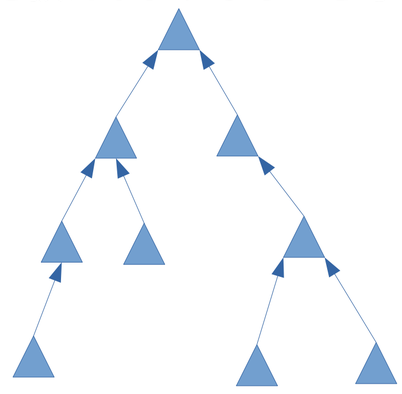
WSN向けのマクロプログラミング言語、Regimentをベースとした WSAN向けマクロプログラミング言語をBrigadeを提案する。

一つのBrigadeプログラムから、コンパイラによってセンサーノード、アクターノードそれぞれに向けたプログラムが生成される。

時間と共に変化する値の集合をRegionと呼ぶ。各センサーノードが計測する情報をまとめてRegionとみなし、アクターノードが収集する。各アクターノードが持つ値も同様にまとめてRegionとみなし、集約を行えるようにする。

アクター・アクター間協調の方法として、近隣のアクターとデータを交換し、自身の動作を決める方式と、全てのアクターのデータをRegionとして一つのアクターに集約し、その情報をもとに各アクターの挙動を決める方法の二つを扱えるようなものを考える。





例：火災検知システム

右の図では両方のアクター間協調方法を用いた火災検知システムの例を記す。

センサー：

- 温度計と煙探知器の値をアクターに送信

アクター：

- センサーの値から火事かどうかを判定
- 火災が発生している付近のアクターのスプリンクラー作動
- 全てのアクターで警報を鳴らす

```
fun read(sensor) {
  (sense("temp", sensor), sense("smoke",
sensor))
}


fun on_fire((t, s)) {
  t > TEMP_THRESH && s > SMOKE_THRESH
}


fun read_actor(actor) {
  (rfilter(on_fire, rmap(read,
get_region(actor))), actor)
}


fun sprinkler((detects, actor)) {
  detect = rfold((||), false, khood(1, actor,
count(detects) > 0))
  actuate(actor, "sprinkler", detect)
}


detects = rmap(read_actor, actors)
```

```
riter(sprinkler, detects)


fire = rfold(fun((x, (y, _))) {x || count(y)
> 0}, false, detects)


riter(fun(actor) {actuate(actor, "alarm",
fire)}, actors)
```

- Region(a)
  型aの時変値の集合
  各Regionにはアンカーと呼ばれるrfoldの集約先が設定される
- actors :: Region(ActorNode)
- rmap :: (a→b)→Region(a)→Region(b)
- rfilter :: (a→bool)→Region(a)->Region(a)
- rfold :: (a→b→a)→a→Region(b)→a
- riter :: (a→())→Region(a)→()
  それぞれregionに対するmap, filter, fold, iter操作
- get_region :: ActorNode→Region(SensorNode)
  アクターに属するセンサーのRegionを返す
  khood :: Int→ActorNode→a→Region(a)
  まずkホップ近隣のアクターノードにaのデータを送信し、次にkホップ近隣が同じkhoodオペレータを用いて送信したaのデータを受け取りRegionとして返す
  get_region, khood共にアンカーはRegionを生成するActorNode

コンパイル手法

Regimentのコンパイルにはネットワーク視点からノード視点へのプログラムの変換を行う過程がある。Brigadeではこの過程でセンサー向けプログラムとアクター向けプログラムを分離することで、各ノードに対応したプログラムを生成する。

- Stream(a)
  型aの時変値
- smap, sfilter, sfold, siter
  Streamに対するmap, filter等の操作
- aggr, emit
  ノード単位のデータ収集、排出
- SENSOR, ACTOR
  送信、受信先のノード種類

	Brigadeコード	制御フローへの変換		ノード視点のオペレータへの変換	
センサーとアクターそれぞれのコードの生成	rfilter(f_1, rmap(fun(actor) { rmap(f_2, get_region(actor)) }, actors))	センサー	get_region → rmap(f_2) → ACTOR	timer → smap(f_2) → ACTOR	
		アクター	actors → rfilter(f_1)	SENSOR → sfilter(f_1)	
近隣のアクターとの協調	rfold(f, c, khood(1, actor, x))	khood(1, actor, x) → rfold(f, c)		emit → aggr(f, c)	
一つのアクターへデータを送信	x = rfold(f_1, c, some_region) riter(f_2)	some_region → rfold(f_1, c)		ルートノード以外	SENSOR → emit
				ルートノード	aggr(f_1, c) → ACTOR
		actors → riter(f_2)		ACTOR → siter(f_2)	

今後の展望

各センサーがデータを送信するアクターの割当方法の考案

Brigadeコンパイラの実装

Brigadeプログラムの実装と評価（特にコンパイル後のプログラムサイズ、使用メモリ量、ノード間通信量）

関連研究

Regiment[R.Newton et al. 2007]：WSN向け関数マクロプログラミング言語

SOSNA[M.Karpiński et al. 2008]：WSAN向けマクロプログラミング言語

センサーとアクターの区別をしておらず、アクター間における通信が非効率的

PICO-MP[N.Dulay et al. 2018]：WSAN向けマクロプログラミングミドルウェア