

# Hormone-Inspired Adaptive Communication and Distributed Control for CONRO Self- Reconfigurable Robots

Author: Wei-Min Shen, Behnam Salemi, and Peter Will, Member, IEEE

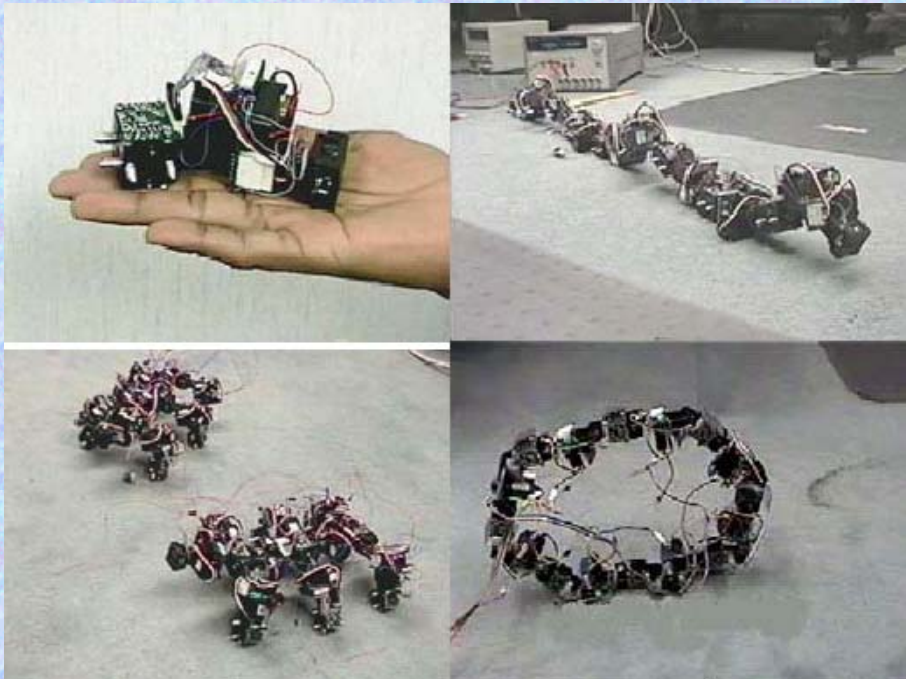
Sponsor: DARPA/ MTO and AFOSR

Published: *IEEE Transactions on Robotics and Automation*, 18(5): 700-712,  
October 2002.

Electronic version: [http://www.isi.edu/conro/papers/ieee\\_tra\\_2002.pdf](http://www.isi.edu/conro/papers/ieee_tra_2002.pdf)

Presenter: Maureen Chandra, 27 February 2003

# Self-Reconfigurable Robots



Self-reconfigurable robots are made of autonomous modules that can connect to each other to form different configurations.

-) Autonomous = has its own controller, communicator, power source, sensors, actuators, and connectors.

# Issues addressed

1. How modules in these robots communicate with each other when connections between them may be changed dynamically and unexpectedly (thus changing their communication routing)
2. How these physically coupled modules collaborate their local actions to accomplish global effects such as locomotion and reconfiguration.

## Things to know:

Coordination between these modules need to be dynamic, asynchronous, scalable and reliable.

They cannot communicate using named addresses (such as in the Internet) since dynamic change to the topology of network requires continually determining the address and computing the route.

# Approach to the problems

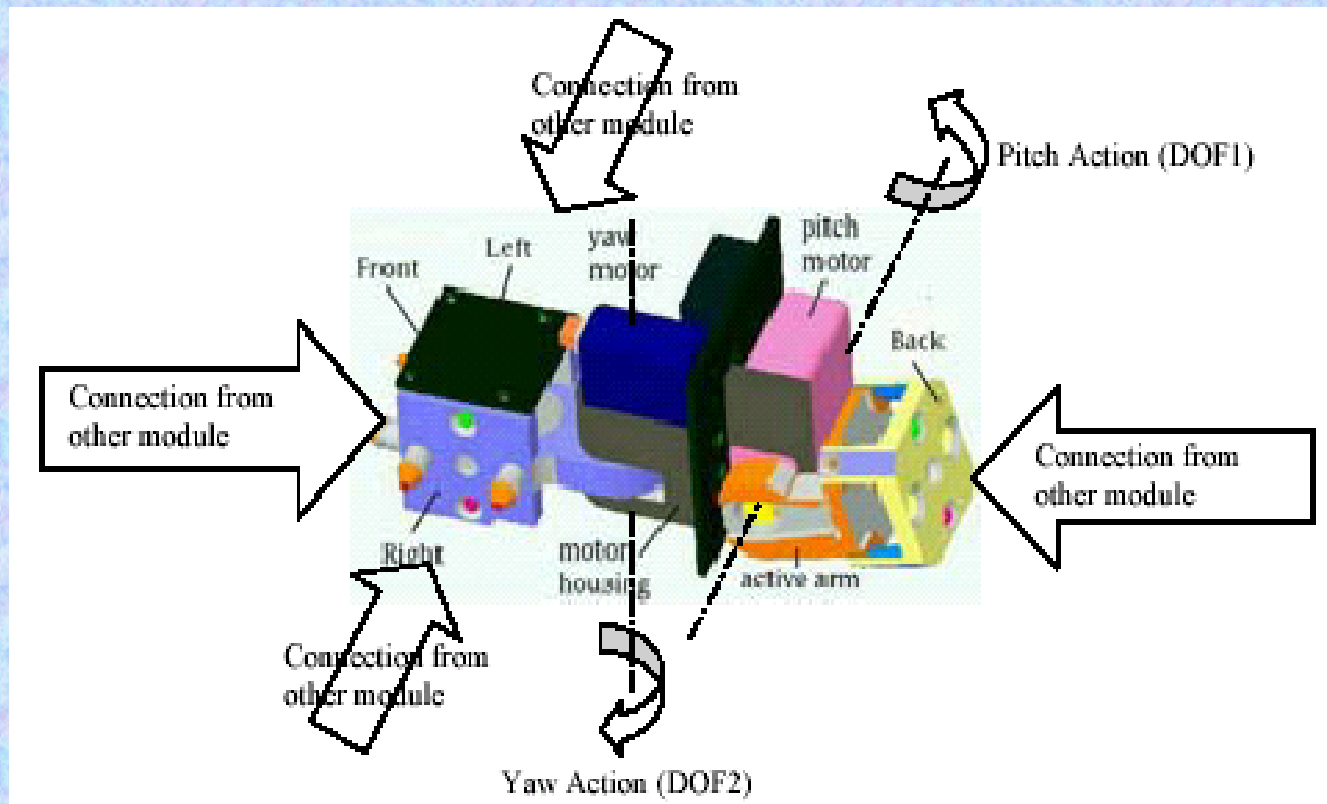
- Concept of *hormones*: act simultaneously without interfering with each other, and each only affect specific target sites.
- Using this concept, a “hormone” signal can be propagate through the entire network of modules and cause different modules to react differently based on their local information.
- The differences between a hormone signal and content-based message are: 1) it has no specific destination, 2) it propagates through the network, 3) it may have a lifetime, 4) it may trigger different actions for different receivers.

Using this idea, two types of protocols were designed:

1. Adaptive Communication (AC) protocol
2. Adaptive Distributed Control (ADC) protocol

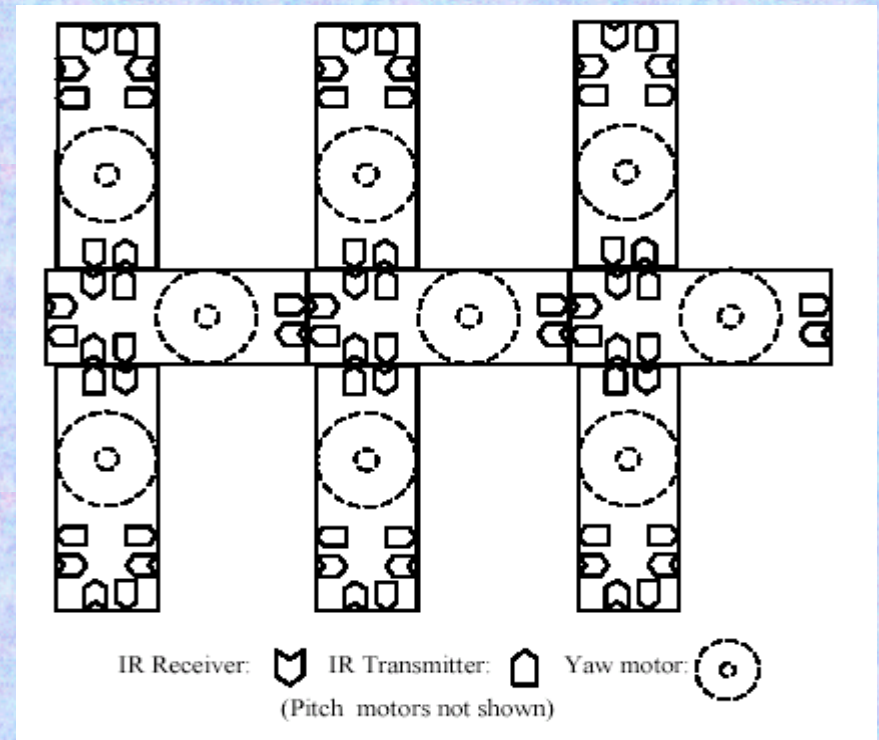
# Self-Reconfigurable Modules and Networks

The schema for CONRO self-reconfigurable robots, the ones used in these experiments

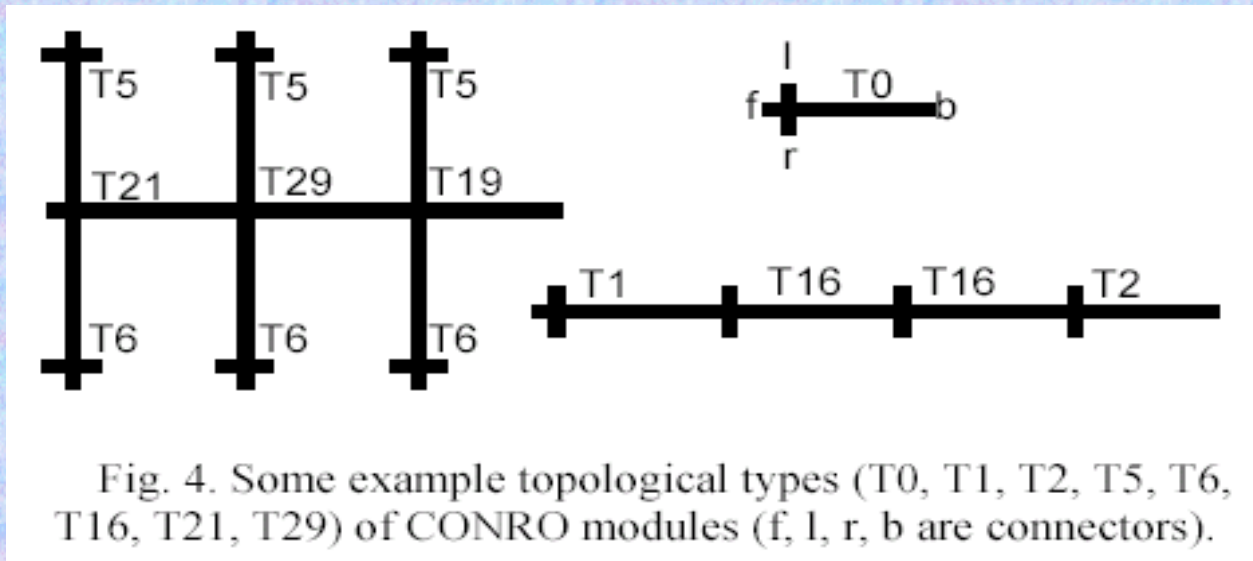


# Self-Reconfigurable Modules and Networks

- The back connector is a *female* connector and the front, left and right connectors are *male* ones.
- When the female and male connector are joined together, it is called *active link*. Connected modules are called *neighbors*.
- CONRO modules communicate with each other through active links. Each connector has an *infrared transmitter* and an *infrared receiver*.



# Representation of Local Topology



Local topology of a CONRO module in a self-reconfigurable network is based on how its connecteds are connected to the connectors of its neighbor modules.

# Representation of Local Topology

TABLE I: THE LOCAL TOPOLOGY TYPES OF CONRO MODULES

	This Module					This Module				
	<i>b</i>	<i>f</i>	<i>r</i>	<i>l</i>	Type	<i>b</i>	<i>f</i>	<i>r</i>	<i>l</i>	Type
Connected to other modules					T0	<i>f</i>	<i>b</i>			T16
	<i>f</i>				T1	<i>f</i>		<i>b</i>		T17
		<i>b</i>			T2	<i>f</i>			<i>b</i>	T18
			<i>b</i>		T3		<i>b</i>	<i>b</i>	<i>b</i>	T19
				<i>b</i>	T4	<i>f</i>	<i>b</i>	<i>b</i>		T20
	<i>l</i>				T5	<i>f</i>		<i>b</i>	<i>b</i>	T21
	<i>r</i>				T6	<i>f</i>	<i>b</i>		<i>b</i>	T22
		<i>b</i>	<i>b</i>		T7	<i>l</i>	<i>b</i>	<i>b</i>		T23
			<i>b</i>	<i>b</i>	T8	<i>l</i>		<i>b</i>	<i>b</i>	T24
		<i>b</i>		<i>b</i>	T9	<i>l</i>	<i>b</i>		<i>b</i>	T25
	<i>l</i>	<i>b</i>			T10	<i>r</i>	<i>b</i>	<i>b</i>		T26
	<i>l</i>		<i>b</i>		T11	<i>r</i>		<i>b</i>	<i>b</i>	T27
	<i>l</i>			<i>b</i>	T12	<i>r</i>	<i>b</i>		<i>b</i>	T28
	<i>r</i>	<i>b</i>			T13	<i>f</i>	<i>b</i>	<i>b</i>	<i>b</i>	T29
	<i>r</i>		<i>b</i>		T14	<i>l</i>	<i>b</i>	<i>b</i>	<i>b</i>	T30
<i>r</i>			<i>b</i>	T15	<i>r</i>	<i>b</i>	<i>b</i>	<i>b</i>	T31	

# Adaptive Communication Protocol

- Adaptive Communication Protocol ( AC) is used to continually discovering the network topology and ensure adaptive communication.
- The main procedure from its program on the right, is a loop of propagating “probe” hormones between neighbors, and selecting and executing local actions based on these messages.
- A probe: a special type of hormone that is used for continuously discovering and monitoring local topology
- LINK[1,...,C]: the status variables for the connectors (i.e., the local topology), and their initially values are nil.

```
Main()
LocalTimer = 0;
Loop forever:
  For each connector c=1 to C, insert [probe,_,c,_] in OUT;
  For each received hormone [type, data, sc, rc] in IN, do:
    { LINK[rc] = sc;
      If (type ≠ probe) then
        SelectAndExecuteLocalActions(type, data);
        PropagateHormone(type, data, sc, rc);
    }
  Send();
  LocalTimer = mod(LocalTimer+1, MaxClock);
End Loop.

SelectAndExecuteLocalActions(type, data)
{ // For now, assume that when LocalTimer=0, a module will
  // generate a test hormone to propagate to the network
  // Other possible local actions will be introduced later.
  If LocalTimer=0, then for c=1 to C, do:
    Insert [Test, 0, c, nil] into OUT;
}

PropagateHormone(type, data, sc, rc)
{ For each connector c=1 to C, do:
  If LINK[c]≠0 and c≠rc, then
    { Delete [probe, *, c, *] from OUT;
      Insert [type, data, c, nil] into OUT; // propagation
    }
}

Send()
{ For each connector c=1 to C, do:
  get the first message [type,*,c,*] from OUT,
  Send the message through the connector c;
  If send fails (i.e., time out), LINK[c] = 0.
}
```

# Adaptive Communication Protocol

## Important properties in AC protocol:

1. All modules can adapt to the dynamic topological changes in the self-reconfigurable network and discover their local topology in a time less than two cycles of the main loop.
2. If the network is acyclic graph, the it guarantees that every non-probe message will be propagated to every module in the network once and only once.

# Adaptive and Distributed Control Protocol

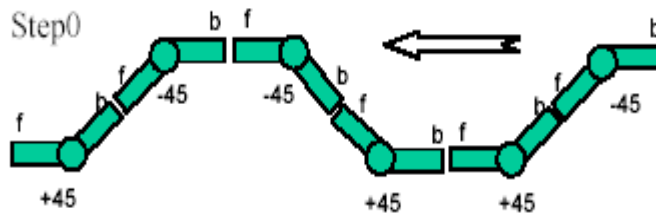


Fig. 6. A caterpillar (or nessie) movement  
(b and r are connectors, and +45 and -45 are DOF1)

TABLE 2: THE CONTROL TABLE FOR THE CATERPILLAR MOVE

Step	Module ID for DOF1 actions					
	M1	M2	M3	M4	M5	M6
0	+45°	-45°	-45°	+45°	+45°	-45°
1	-45°	-45°	+45°	+45°	-45°	-45°
2	-45°	+45°	+45°	-45°	-45°	+45°
3	+45°	+45°	-45°	-45°	+45°	+45°

The control table used here needs to be changed every time the robot configuration is changed.

However, realizing that each module go through the same sequence of motor actions (+45, -45, -45, +45), ADC is designed in order to solve the problem.

# Adaptive and Distributed Control Protocol

TABLE 3: THE RULEBASE FOR THE CATERPILLAR MOVE

Module Type	Local Timer	Received Hormone Data	Perform Action	Send Hormone
T1	0		DOF1=+45	[CP, A, b]
T1	(1/4)*MaxClock		DOF1=-45	[CP, B, b]
T1	(1/2)*MaxClock		DOF1=-45	[CP, C, b]
T1	(3/4)*MaxClock		DOF1=+45	[CP, D, b]
T16,T2		A	DOF1=-45	[CP, B, b]
T16,T2		B	DOF1=-45	[CP, C, b]
T16,T2		C	DOF1=+45	[CP, D, b]
T16,T2		D	DOF1=+45	[CP, A, b]

The ADC protocol is an AC protocol with an extended `SelectAndExecuteLocalActions()` procedure and a RULEBASE that is used to select actions.

The selection process is based on: 1) local topology information(LINK), 2) the local state information(timer), 3) the received hormone messages

Important properties in ADC protocol: 1) distributed( no “brain) and fault-tolerant, 2) collaborative behaviour (no unique ID), 3) asynchronous coordination, 4) scalability

There are other locomotion examples of the ADC protocol including a legged robot and rolling track. They are also discussed in the paper.

# Distributed Control of Cascade of Actions

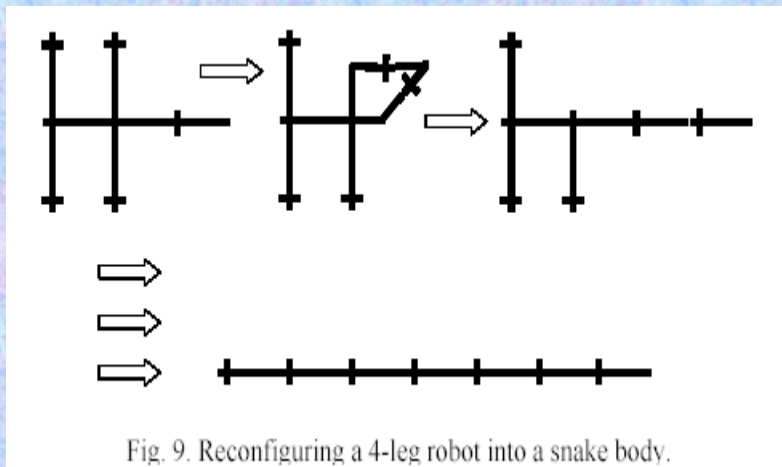


TABLE 6: THE HORMONE ACTIVITIES FOR CASCADE ACTIONS

Hormones	Actions
LTS	Start the reconfiguration
$RCT_1, RCT_2, RCT_3, RCT_4$	Legs are activated to generate RCTs
$TAR, RCT_2, RCT_3, RCT_4$	The tail accepts a RCT, and leg1 stops $RCT_1$
$ALT, RCT_2, RCT_3, RCT_4$	The tail and leg1 perform the assimilation process
$TAR, RCT_2, RCT_4$	The new tail accepts a RCT, and leg3 stops $RCT_3$
$ALT, RCT_2, RCT_4$	The tail and leg3 perform the assimilation process
$TAR, RCT_2$	The new tail accepts a RCT, and leg4 stops $RCT_4$
$ALT, RCT_2$	The tail and leg4 perform the assimilation process
TAR	The new tail accepts a RCT, and leg2 stops $RCT_2$
ALT	The tail and leg2 perform the assimilation process
$\emptyset$	No more RCT, and end the reconfiguration

Control Cascade of actions is where actions are organized in a hierarchical structure and a single action in a higher-level can trigger a sequence of lower-level actions.

LTS = Legs To Snake, RCT = Request to Connect to the Tail, TAR = Tail Accept Request, ALT, Assimilate Leg into Tail.

# Related Work

Some of the related work for distributed control includes: -) the series of control algorithms proposed by Murata et al for self-assembly and self-repairing robots, -) the role-based control method by Stoy et al, -) a goal-ordering based approach by Yim et al and -) an automata-based approach by Butler et al.

This paper is different from previously proposed method in several ways:

1. A module selects actions based on multiple sources of local information
2. The local topology does not only know that its connector is connected to a neighbor, but also know the neighbor's connector type.
3. It can deal with both locomotion and reconfiguration using the same framework.
4. It has wider application scope and thus can support modules that have internal deforming actions such as pitch, yaw and roll.

# Results

Tested in two sets of experiments, the real CONRO modules and the simulation called Working Model 3D, these results were achieved:

- The experiment on adding and taking away modules in snake configuration shows that the ADC protocol is robust to changes in the length.
- The experiment on flipping over the snake shows that modules can automatically generate hormones when they receive appropriate environmental stimuli from their local sensors.
- The experiment on removing one or more legs on legged configuration show that the protocol still work nevertheless. Even with one leg, the robot would still attempt to walk
- The experiment on simulating random message losses in the rolling track configuration shows that the chance of failing to resume rolling after losing messages in between is very low.

# Contribution

Like all other methods, this one has its weaknesses, including:

- Difficulty in developing the appropriate RULEBASE for a particular global behaviour, especially complex ones. It is still an open problem how to develop these rules automatically
- If there are delays in the communication, the system may behave erratically. A possible solution proposed is to adjust local timers to compensate the delay

Nevertheless, this paper contributes a new method that can support many unique features of self-reconfigurable robots, which include:

- Adaptive communication in dynamic network
- Decentralized and distributed control for collaboration among autonomous modules
- On-line reconfiguration
- Scalability to larger and multiple robotic systems

Questions?