

Resource Analysis for Hume Box Compositions

*K. Hammond, C. Herrmann:
University of St Andrews*



*SEN018: Resource-driven reconfiguration of AV systems
in collaboration with SER015, Waterfall Solutions Ltd*

*SEAS-DTC Annual Technical Conference
Edinburgh, 7/8 July 2009*

SEN018/SER015

- **Joint inter-theme research programme**
- **Waterfall Solutions will (SER015)**
 - enhance the *Cerberus* infrastructure
 - simulate reconfigurable systems of multiple assets involved in search activities
 - develop new search patterns
- **The University of St Andrews will (SEN018)**
 - provide advanced resource models and analyses for systems of multiple assets that can be used to support autonomous reconfiguration





SEN018/SER015 Objectives

- **Aims to demonstrate autonomous reconfiguration capabilities under changing resource demands**
 - system comprising multiple cooperating AV assets
 - assets may be of different types (land, sea and air)
- **Reconfiguration will exploit resource models to deal with changing demands**
 - real-time deadlines
 - power consumption
- **Reconfiguration will be goal-directed**
 - to allow e.g. optimisation of search patterns under resource requirements



Potential Military Value

- **Increased Platform predictability/reliability**
 - platform will execute commands in guaranteed time / power consumption
 - platform will not fail due to underestimation resource requirements
- **Certiability**
 - required properties can be formally & independently verified
- **Enhanced capability/survivability**
 - defunct platforms due to external impact will be replaced by reconfiguring the others
- **Supports rapid prototyping of military software**
 - more design choices can be explored
 - faster systems development/deployment
 - higher likelihood that systems requirements are met in practice



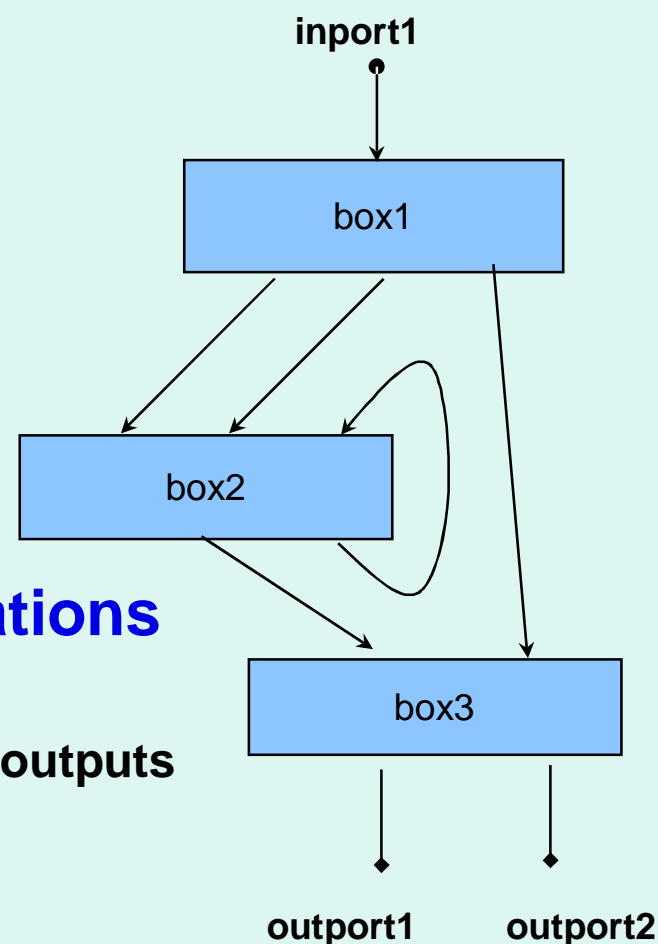
Hume Language Structure

- **Boxes structure processes**

- Static process network
- **Asynchronous** communication
- **Stateless** finite automata

- **Functions structure computations**

- Purely functional notation
- Pattern-matching relates inputs to outputs through functional expressions





Using the Hume Program

- **Simulation**
 - each Hume program can execute the mission computer of one simulated platform to obtain realistic results
 - controlled and observed e.g. by Cerberus (Waterfall Solutions Ltd.)
- **Potential for real system**
 - I/O channels to be connected to wireless communication devices
 - pseudoboxes to be replaced by real devices

Determining Resource Bounds

1. Build Formal operational semantics

- captures low-level information (PowerPC 603e)



T_{plus} = 1
T_{push} = 3
...

2. Build mathematical models of resource usage

- relate programs to resource use
- formal models of complex program structures, real-time constructs
- metrics: execution time, stack high watermarks, memory allocations/deallocations
- provable *bounds* on resource use

$$\mathcal{V}, \eta \vdash \frac{t}{t'} \frac{p}{p'} \frac{m}{m'} e \rightsquigarrow \ell, \eta'$$

$$T_{\text{init}} = T_{\text{call}} + 5 \cdot T_{\text{pushvar}} + 3 \cdot T_{\text{mkint}} + T_{\text{mkvec}}(2) + \dots + T_{\text{createframe}} + T_{\text{matchrule}} + \dots$$

3. Construct static analyses

- based on mathematical models

4. Analyse AV program source

- extract bounds on resource use

```
case asp of
  (Position xs) -> (PNORM myId send 10
    (Response (Position myPos)), (myId, myPos, myPower, 0), *)
| (Power xs) -> (PNORM myId send 10
  (Response (Power myPower)), (myId, myPos, myPower, 0), *)
| (Target (JF angle)) -> (*, (myId, myPos, myPower, 0), angle)
```

5. Compare bounds with measured usage

- time and space on actual platform

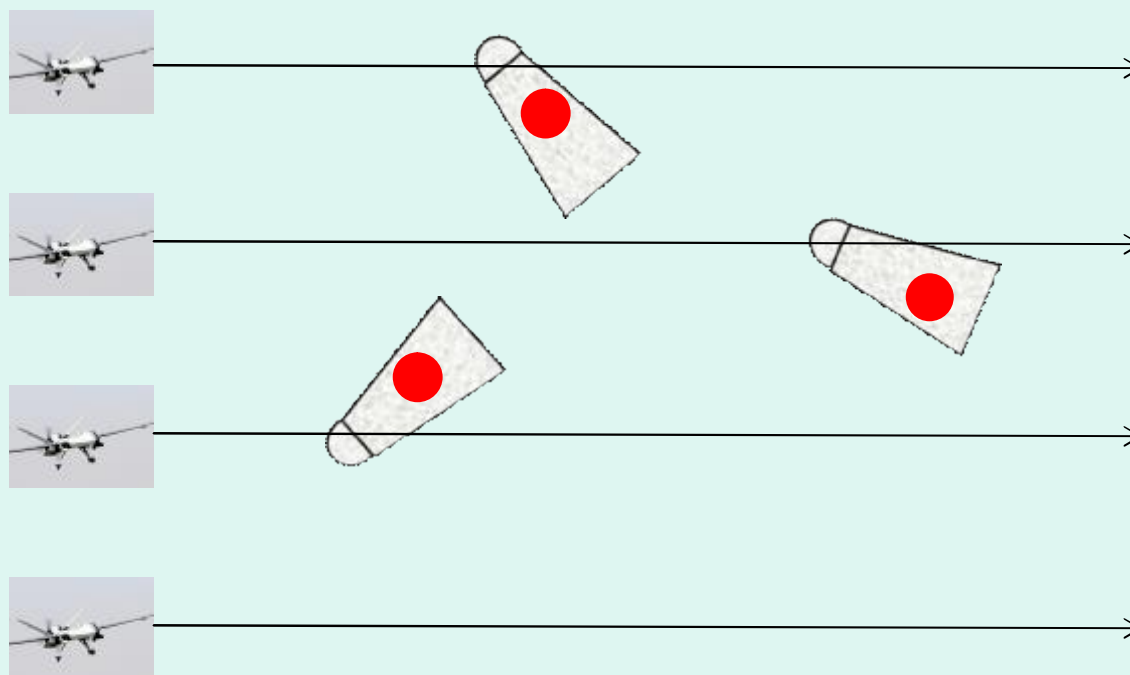
6. Adapt AV software if necessary to improve bounds





Application Scenario 1





- Airborne-based tracking of ground objects
 - a flock of UAVs traverses an area for targets (●)

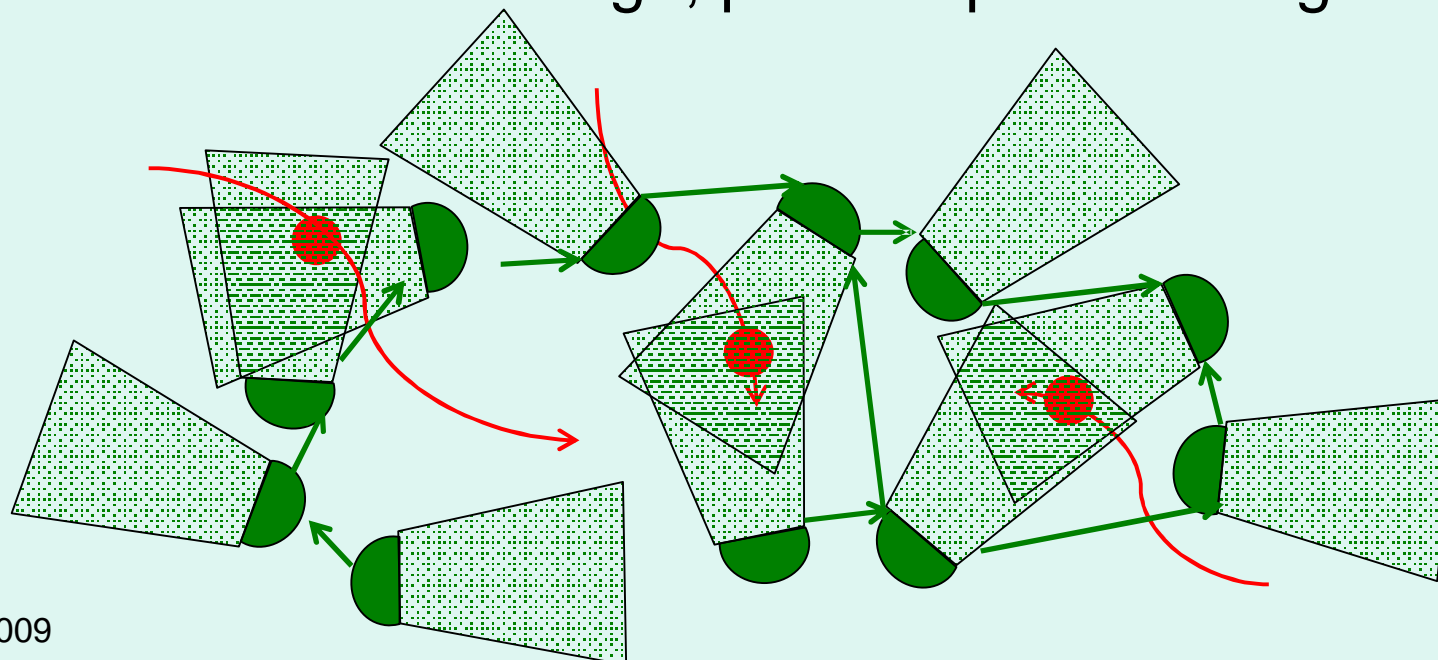


Application Scenarios

1. Airborne-based tracking of ground objects
 - a flock of UAVs traverses an area
2. Ground-based tracking of moving ground objects
 - a) using stationary platforms
 - operating sensor
 - exchanging information about target movements
 - b) using AVs
 - changing own position to increase coverage






Scenario 2a (Setup)

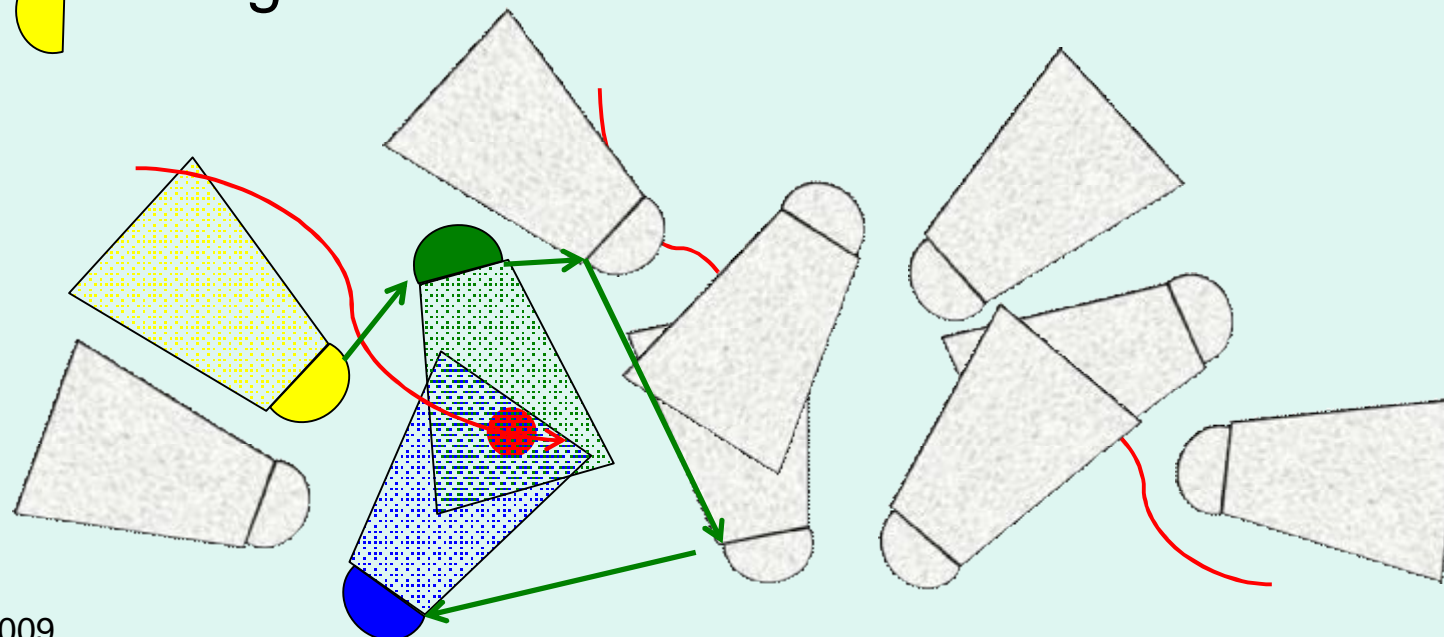
- Stationary, ground-based stations  
 - distributed across an area to be observed
 - localisation and ranging of moving objects 
 - wireless data exchange, point-to-point routing 







Scenario 2a (Area coverage)

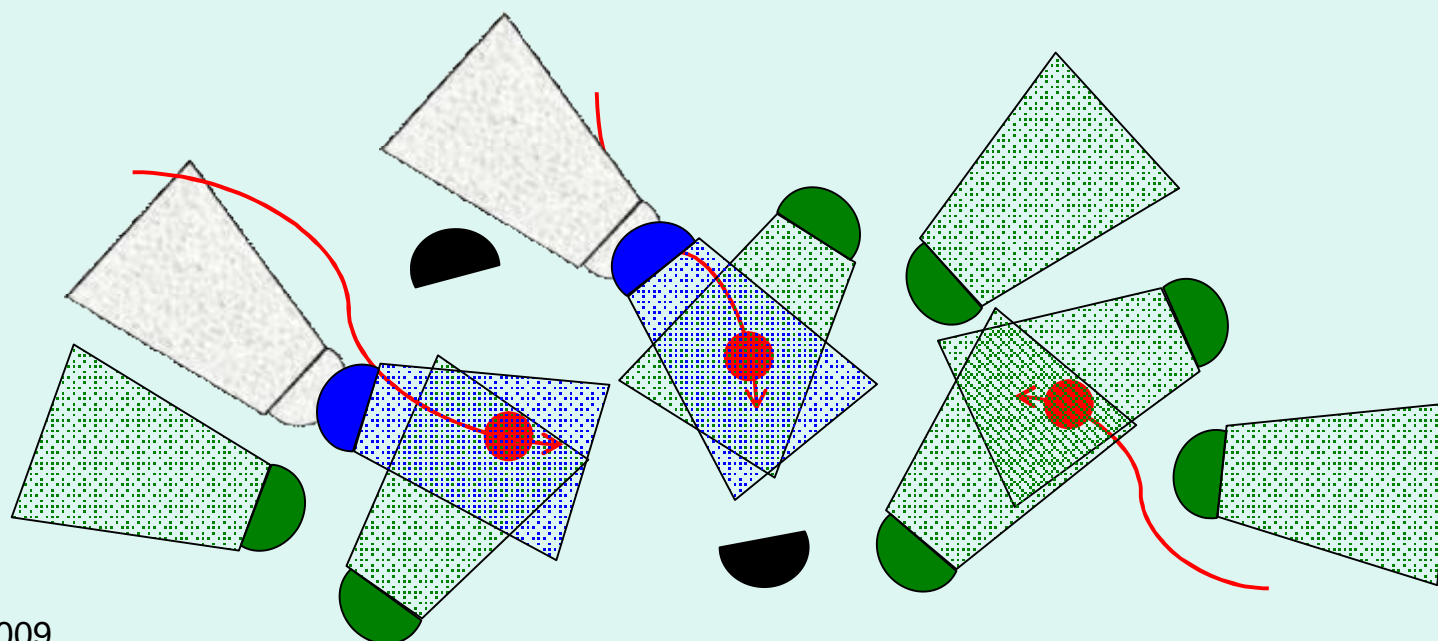
- Object has moved out of range of 
 - other platform  keeps track and notifies 
 -  changes tracking direction
 -  changes direction to cover different area





Scenario 2a (Reconfiguration)

- Ex.: two tracking assets defunct 
 - other platforms exchange information
 - two trackers  change field of view to cover objects





Reconfiguration

- Situations
 - tracked object moves out of range of platform, platform can notify neighbours
 - platform got defunct, this must be detected by others
 - energy level drops: platform only performs routing
- Treatment of reconfiguration
 - strategy part of each AVs control component
 - reconfiguration has known resource demand
 - reconfiguration uses information of resource usage
 - separation of concerns: intra-asset, inter-asset



Current Focus: Single Asset

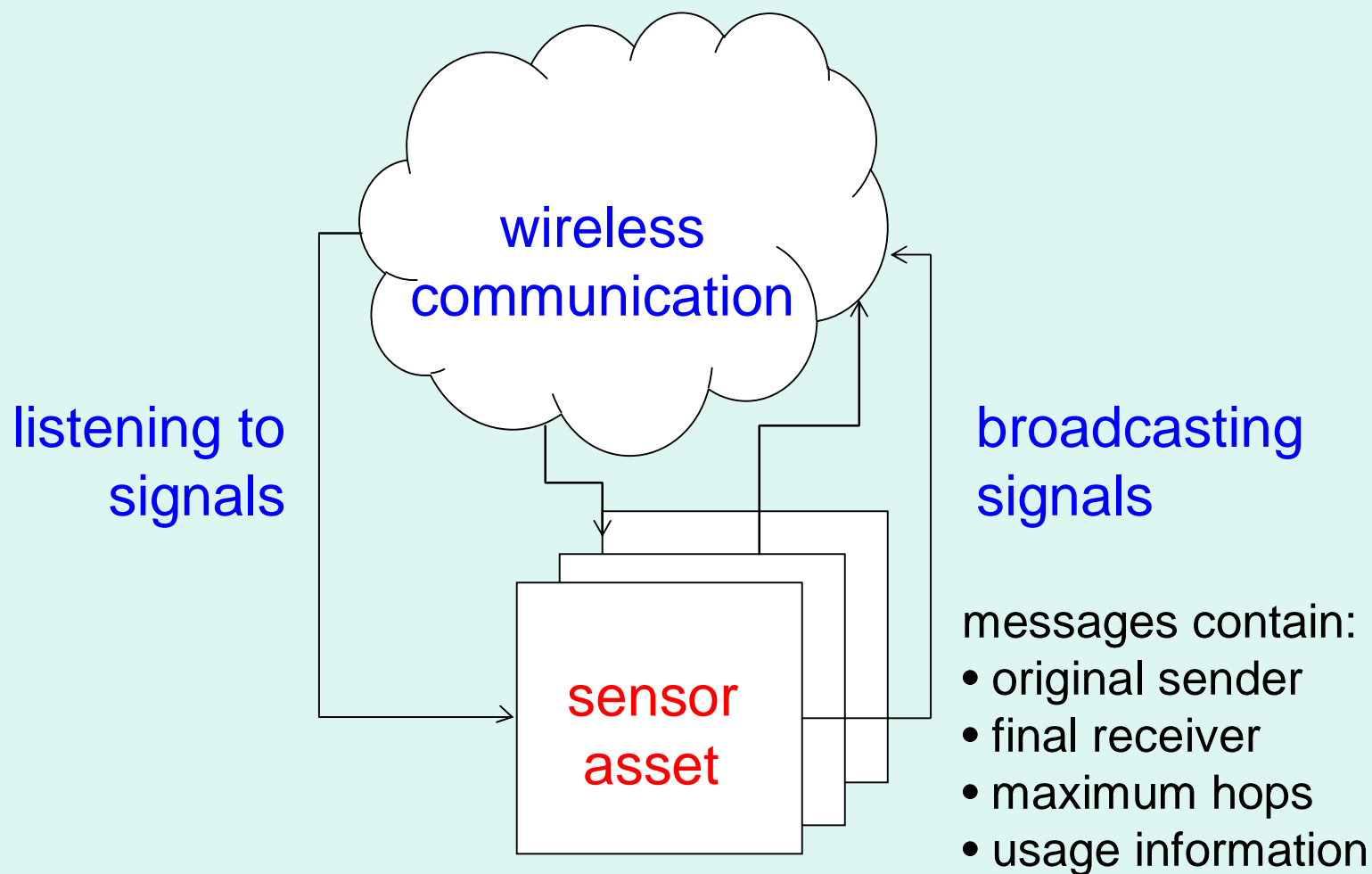
- Basis for reconfiguration strategy
- Resource information for all modes
 - obtained at design time by analysis
- Different Abstraction levels

Levels of Abstraction

1. Machine Level Analysis
 - analysis of assembly of each basic Hume instruction
2. Operational Semantics, “Amortised Analysis”
 - Hume expressions and box branches
3. Analysis of compositions (Hume boxes, external devices)
 - Involves multiple box executions
 - includes receiving command, sensing, calculating and returning answer
4. Planning the collaborative action including reconfiguration
 - planning algorithms based on costs of point 3.



Single Asset: Role in the System

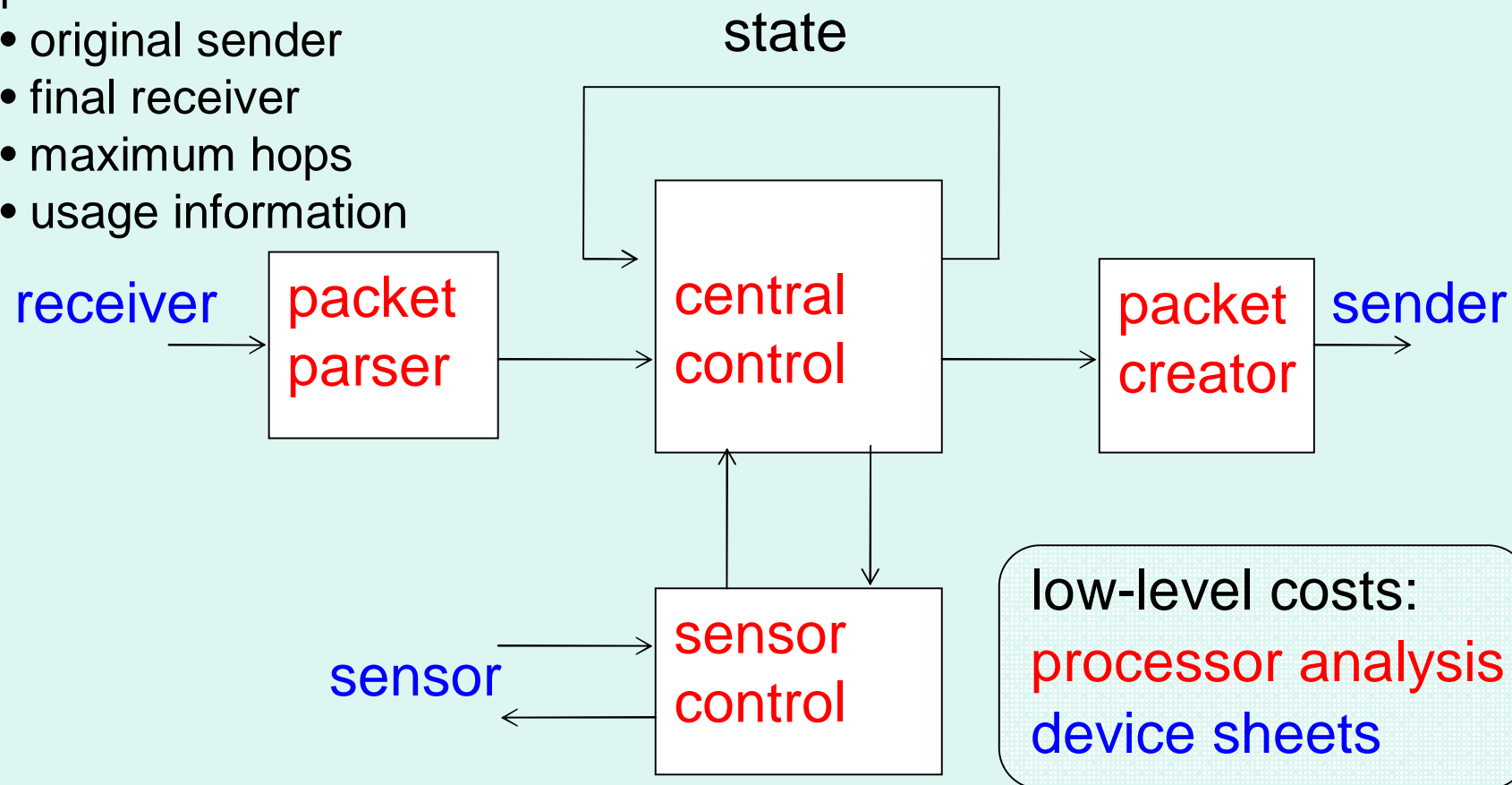




Single Asset Structure

packets contain:

- original sender
- final receiver
- maximum hops
- usage information





Usage Information Types

- packet types: request / response
- if request packet & myId=receiver
 - then send back response packet
 - else broadcast packet to reach more stations
- aspects to deal with
 - query state information (AV position, energy)
 - target detection in direction
 - several scheduling cycles: receive request, ..., operate sensor, change state, ..., send response
 - time/power encoded in the program for profiling

Receiving Request

```
(PNORM send recv hops (Request asp (myId,myPos,myPower,xs), *) ->
  (if recv==myId
    then case asp of
      (Position xs) -> (PNORM myId send 10
        (Response (Position myPos)),
        (myId,myPos,myPower,0), *)
    | (Power xs) -> (PNORM myId send 10
        (Response (Power myPower)
        (myId,myPos,myPower,0), *)
    | (Target (JF angle)) -> (*, (myId,myPos,myPower,0)
        angle)
        [wire to sensor]
```

Response to Tracking

```
(* , (myId, myPos, myPower, requestFrom),  
  (angle, strength, dt, dp))           [wire from sensor]  
  -> (PNORM myId requestFrom 10  
      (Response (Target (if strength<1.0  
                        then NoF else JF angle))),  
      (myId, myPos, myPower-dp, 0), *)
```

- **angle**: the direction in which the target was located
- **strength**: the intensity of the radar response, if less than 1.0 no target is reported
- **dt / dp**: the time/power consumed by the sensor (a formal hook for analysis)

Analysis Procedure

- Obtain resource consumption for each mode
 - cheap operations: status query, packet routing
 - high resource consumption: sensor operation
- Calculate resource usage for each packet type
 - e.g.: tracking involves
 - receiving packet (control box)
 - operating the sensor (sensor box)
 - interpreting result, sending reply (control box),
∅ the control box is executed in two different modes,
semantic analysis reveals the dependencies



Analysis Results for Time

Box	machine cycles	microseconds
central control	111672	335
scan input	$70652+25571*n$	$212+77*n$
sensor control	110870	333

- n: length of the input command string
- times for central control can be further discriminated
- several other boxes involved
- unpacking not yet costed (not linear)
- expected CPU times for simple services: 10-100 milliseconds
(like forwarding packets or delivering sensor information)

Integration of Hardware Costs

- Example: compositional analysis obtains
 - one execution of each Hume Boxes A and B ($a+b$)
 - $n+4$ executions of Hume Box C $((n+4)*c)$
 - external devices
 - receiver once (d)
 - sender once (e)
 - sensor n times turn: ($n*f$)
 - sensor n times pulse/scan: ($n*g$)
- total: $\text{cost}(n) = (a+b+4c+d+e) + n*(c+f+g)$

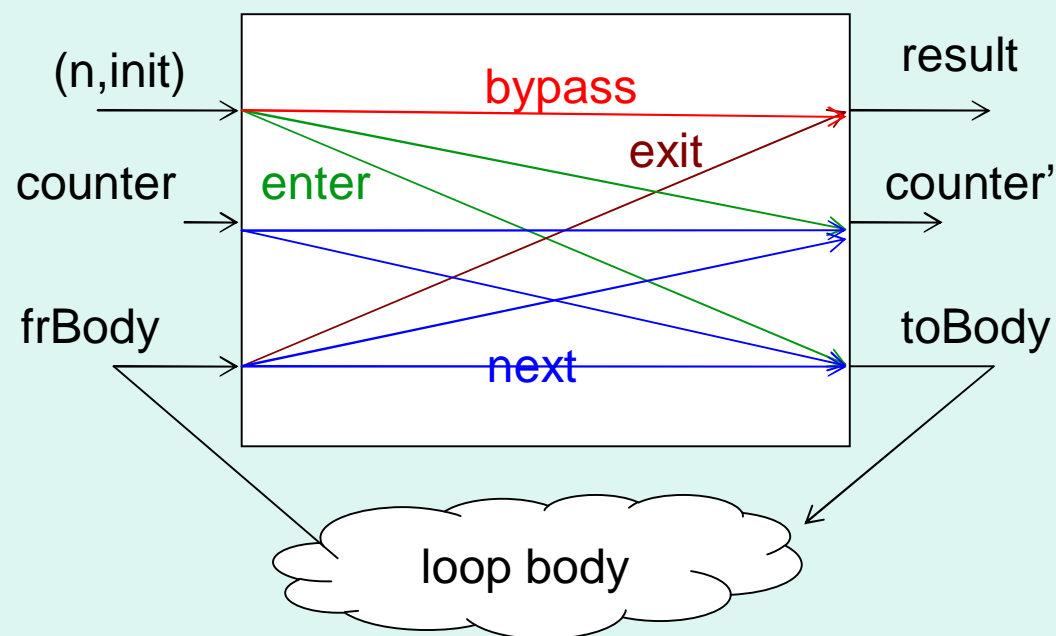
Dealing with Variables

- Challenge: different values can trigger different modes or numbers of box repetitions
- Potential approaches
 - separate parameters which impact control flow and perform a case distinction on them (-> cost table)
 - design patterns (“loop box”) which are analysed once with manual assistance and can be reused in many different situations to implement, e.g., repetition
 - cost function: structure by pattern, arguments by use



Example: Loop Box

- Loop body must only be connected with the loop box
- Costs of **next** and **body** simple (e.g. polynomials)
- Dependent types can enforce proper use





Capabilities

- Precision
 - discriminate different operational modes
 - regard semantic dependencies
- Generic results
 - resource formulae expressed using parameters for
 - iterations
 - sizes of data structures
- Compositional Design and Analysis
 - components have known resource requirements
 - low-level analysis once only for each component
 - integration of properties of external devices