AnonyTL Specification

Dan Peebles, Cory Cornelius, Apu Kapadia, David Kotz, Minho Shin, Nikos Triandopoulos

Dartmouth Computer Science Technical Report TR2010-660

This report describes the AnonyTL tasking language used by the AnonySense system [1, 2].

AnonyTL uses a Lisp-inspired syntax and has declarative semantics. Tasks can specify an identifier, linkability, and when they expire along with a series of statements. Statements encode conditions under which a node should accept or reject a task, what a task should report at what intervals and when the task should stop running. Conditions can involve sensor readings (time and location are considered sensors). There is special syntax for describing location conditions.

1 Grammar

```
task = "(" , s , "task" , s , number , s , [ "linkable" ] , s ,
       "(" , s , "expires" , s , number s , ")" , s ,
       { statement } , ")";

statement = "(" , s , "accept" , s , condition , { condition } , s ")" , space
          | "(" , s , "reject" , s , condition , { condition } , s ")" , space
          | "(" , s , "report" , s ,
              "(" , s , reportable , { reportable } , s , ")" , s ,
              "(" , s , "every" , s , duration, s , ")" , s ,
              condition , { condition } , s , ")", s
          | "(" , s , "quit" , s ,
              "(" , s , "every" , s , duration , s , ")" , s ,
              condition , { condition } , s , ")" , s

condition = "(" , s , comparable binop , s , comparable , s , comparable , s , ")"
          | "(" , s , conditional binop , s, condition , s , condition , s , ")"
          | "(" , s , conditional unop , s , condition , s , ")"
          | "(" , s , numeric binop , s , number , number , s , ")"
          | "(" , s , numeric unop , s , number , s , ")"
          | "(" , s , location condition , s , ")"

location condition = "at" , s , string
                   | "in" , s , "(" , s , { point } , s , ")"
```

```
reportable = symbol , s
           | "(" , s , string , s , string , s , ")" , s

comparable = symbol | attribute | string | number

duration = number , s , "day"
         | number , s , "hour"
         | number , s , "minute"
         | number , s , "second"

point = "(" , s , number , s , number , s , ")" , s

comparable binop = "<=" | ">=" | "!=" | "=" | ">" | "<"

conditional binop = "and" | "or"

conditional unop = "not"

numeric binop = "between"

numeric unop = "before" | "after"

attribute = "@" , symbol

symbol = alpha , { word character }

string = '"' , { ? anything except '"' or newlines ? } , '"'

number = one to nine , { zero to nine }
       | "0"

s = { ? white space characters ? }

alpha = ? alphanumeric characters of any case ?

one to nine = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

zero to nine = "0" | one to nine

word character = alpha | zero to nine
```

2 Semantics

As a declarative language, many of the low-level details are left to the implementation. An AnonyTL program only specifies conditions under which certain behaviors should occur, and the implementation takes care of scheduling to the best of its ability. For example, if a condition depends on the value of a time sensor, the scheduler knows exactly when to fire the behavior. Other sensor types might need a polling infrastructure in the scheduler to check on conditions. We deliberately do not specify details of the scheduler as different platforms will have different resource constraints and hardware architectures, but we do expect an implementation to make a best-effort attempt to trigger behaviors as soon as their conditions can be observed to be satisfied. Individual statements may still provide sampling rate hints for the scheduler through the `every` clause.

Condition clauses in statements are constructed out of predicates on attributes (`attribute` in the grammar) and sensors (`symbol` in the grammar). We define sensors to be any time-varying reading from outside the program. For consistency, we consider time itself to be a sensor, but actual devices might also provide barometers, accelerometers, and other such physical sensors. Attributes are relatively static properties of the node or carrier (e.g., carrier's age group or the device placement on the body). Both sensors and attributes are addressed with string-based keys, but the latter are prefixed with an @ symbol to distinguish them.

The language is intentionally restricted: there are no explicit looping or recursion constructs and it is not Turing-equivalent. Our goal is to provide sufficiently expressive constructs for researchers to collect data but also to allow significant automated reasoning about AnonyTL programs to determine privacy properties.

3 Examples

The following is a simple task that expires at UNIX time 1277364354 (June 24, 2010). It specifies that a node should only accept the task if the carrier is a professor and the sensor is at the hand level, or if the carrier is a student (we use double negation to demonstrate more logical operators). The task then specifies that every minute, if the node's location is in Sudikoff (the mapping of names to locations is handled with a database, or explicit coordinates can be input), the decibel level should be reported along with an arbitrary tag/value pair as an aid to the researcher.

```
(task 1
    (expires 1277364354)
    (accept (or (and (= @carrier "professor") (= @sensor_placement "hand"))
                (not (!= @carrier "student"))))
    (report (decibels ( "tag" "value" ) ) (every 1 minute) (at "sudikoff")))
```

The following, simpler task should only be accepted by devices with an 802.11a/b/g-capable wireless card (the `@wifi` attribute in the `accept` statement). It then asks the devices to report their location (`location` sensor) and the list of visible APs (`aplist` sensor) every minute when the user's location is within a triangle.

```
(task 2
    (expires 1196728453)
    (accept (= @wifi 'a/b/g'))
    (report (location aplist) (every 60 seconds) (in ((1 1) (2 2) (3 0)))))
```

References

[1] Cory Cornelius, Apu Kapadia, David Kotz, Dan Peebles, Minho Shin, and Nikos Triandopoulos. AnonySense: Privacy-aware people-centric sensing. In *Proceedings of the 2008 International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 211–224. ACM Press, June 2008. DOI 10.1145/1378600.1378624.

[2] Apu Kapadia, Nikos Triandopoulos, Cory Cornelius, Dan Peebles, and David Kotz. AnonySense: Opportunistic and privacy-preserving context collection. In *Proceedings of the Sixth International Conference on Pervasive Computing (Pervasive)*, volume 5013 of *Lecture Notes in Computer Science*, pages 280–297. Springer-Verlag, May 2008. DOI 10.1007/978-3-540-79576-6_17.