Why is the Internet of Things (IoT) Definition-less?

Foundations of the IoT

J. Voas

Computer Scientist

US National Institute of Standards and Technology

jeff.voas@nist.gov

j.voas@ieee.org

Question

Where is *the* Definition for IoT?

Wikipedia's Got One

*The **Internet of Things** (**IoT**) is the interconnection of uniquely identifiable embedded computing devices within the existing Internet infrastructure.*

IEEE's IoT Initiative Is Working On One

- from the Initiative's White Paper

- **Small environment scenario:**
  - It's a network that connects uniquely identifiable *"Things"* to the internet.
  - The "Things" have sensing/actuation and potential programmability capability.
  - Information about the "Thing" can be collected.
  - The state of the "Thing" can be changed.
  - From anywhere, at anytime, by anything
- **Large environment scenario:**
  - A self-configuring and adaptive complex network that interconnects *"things"* to the internet through the use of interoperable communication protocol.

InterNational Committee for Information Technology Standards (INCITS) Has One

JTC 1 N 12651 - Text for NWIP ballot on Information technology — Internet of Things — Definition and Vocabulary.pdf modified

Real Answer

Nowhere - No simple, usable, universally-accepted, and actionable definition currently exists.

To address this, I opted to start from the *fundamentals*, i.e., the main *ingredients* that define the *behavior* of the IoT. These fundamentals form the foundation of the IoT.

But, *I do not define IoT*.

Opening Statement

A Network of Things (NoT) employs a mixture of sensing, communication, computation.

A Network of Things (NoT) leads to actionable decisions or predictions. Things may be *private* or *public*. Things may be 3rd party or homegrown.

A Network of Things (NoT) is only one example of a distributed computing system.

The ‘so-called’ Internet of Things (IoT) is one type of a NoT – others exist.

IoT *vs*. NoT

The terms “IoT” and “NoT” are interchangeable - the relationship between NoT and IoT is simple, yet subtle. IoT is an *instantiation* of a network of things, and in particular, IoT has its ‘things’ tethered to the Internet. A different type of NoT, on the other hand, could be a Local Area Network (LAN) of ‘things’ with no access to any ‘thing’ tethered to the Internet. Social media networks, sensor networks, and industrial internet, are variants of NoTs. This differentiation in terminology provides ease in separating out use cases from varying vertical and quality domains (e.g., transportation, medical, financial, agricultural, safety-critical, security-critical, performance-critical, high assurance, to name a few). That proves invaluable, as there is no single static IoT. IoT remains *definition-less*. This is contrary to current discourse.

The Foundation: Eight Primitives

1. Sensor
2. Snapshot (time)
3. Cluster
4. Aggregator
5. Weight
6. Communication channel
7. *e*Utility
8. Decision trigger

Sensor

First Primitive: Sensor – an electronic utility that digitally measures physical properties such as temperature, acceleration, weight, sound, etc. Cameras and microphones are also treated as sensors. The basic properties and assumptions about sensors are:

1. Basic sensors will have little or no software functionality and computing power; more advanced sensors may have software functionality and computing power
2. Sensors will be heterogeneous, from different manufacturers, and collecting any data collectible, with varying levels of data integrity
3. Sensors have operating geographic locations that change. This may occur either by the sensor moving itself such as in the case of a drone, or a sensor that is moved by some other entity
4. Sensors may have owner(s), who will have complete control of the data their sensors collect, who is allowed to access it, and when
5. Sensors have pedigree – geographic locations of origin and manufacturers. Pedigree may be unknown or suspicious
6. Sensors may fail or fail intermittently
7. Most sensors are assumed to be cheap, disposable, and susceptible to wear-out over time; building security into a specific sensor will be rarely cost effective
8. Sensors may return no data, totally flawed data, partially flawed data, or correct/acceptable data
9. Sensors are expected to return data that is in certain ranges, e.g., [1 … 100]. When these ranges are violated, rules may be needed on whether to turn control over to a human or machine if ignoring the out-of-bounds data.
10. Sensor repair is usually handled by replacement

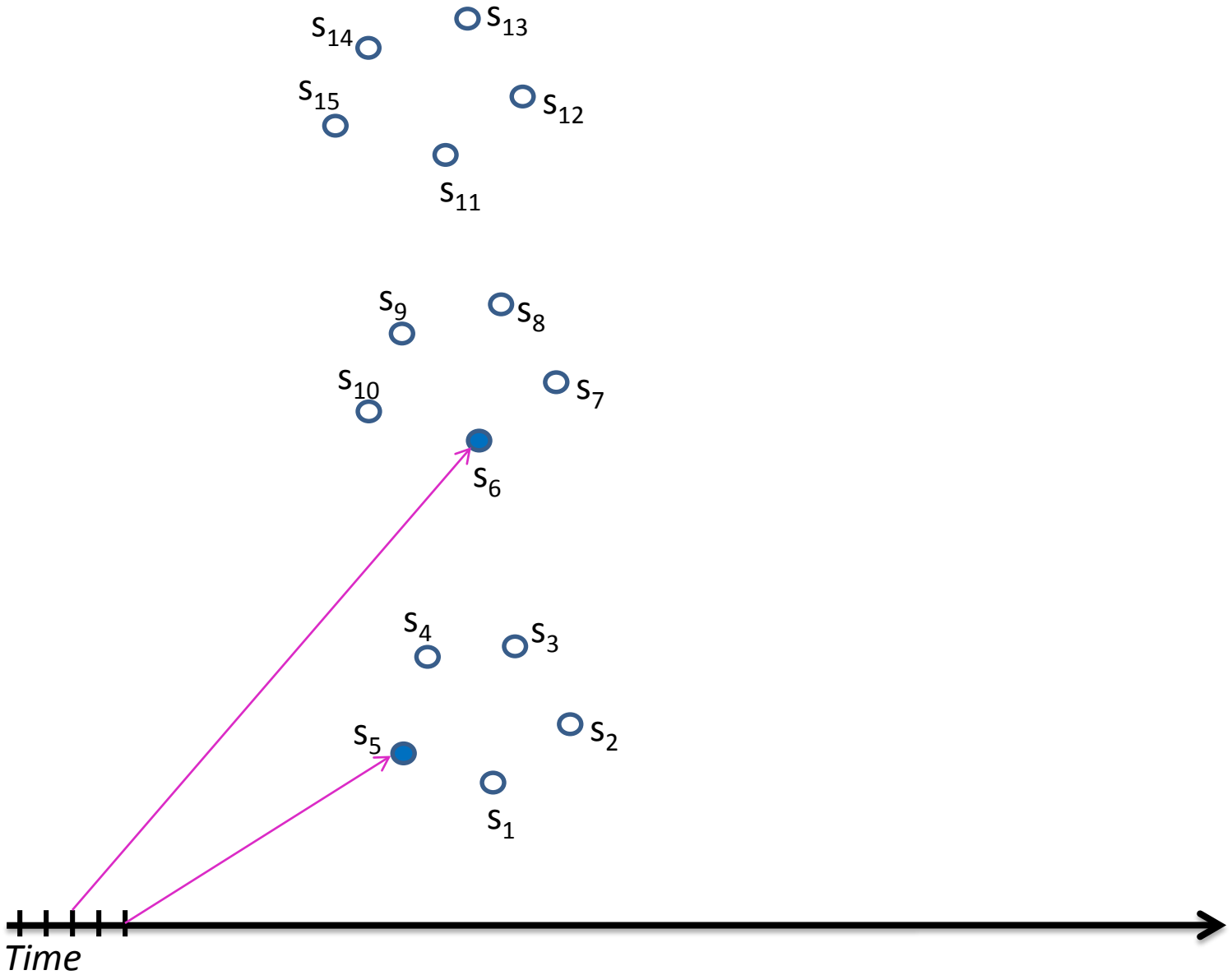
Sensor (Cont'd)

11. Sensors may be acquired off-the-shelf
12. Each sensor can have a level of data integrity ascribed to it
13. Particular sensors may have their data tokenized to void security concerns. For certain application criticalities. If so, tokenization (encryption) is assumed to be correct and immune to compromise
14. Sensors and their data may be leased to multiple NoTs concurrently. A sensor can have one or more recipients' of its data
15. The frequency with which sensors release data impacts the data's currency and relevance
16. Sensor data can be 'at rest' for long periods of time
17. Sensor data can become *stale*
18. A sensor's resolution will determine how much information is provided
19. Security is a concern for sensors if they or their data is tampered with or stolen
20. Reliability is a concern for sensors.

Snapshot

Second Primitive: Snapshot – an instant in *time*. Because a network of things is a distributed computing system, different events, data transfers, and computations occur at different times. Therefore it is necessary to consider *time* as a primitive. The basic properties and assumptions about snapshot are:

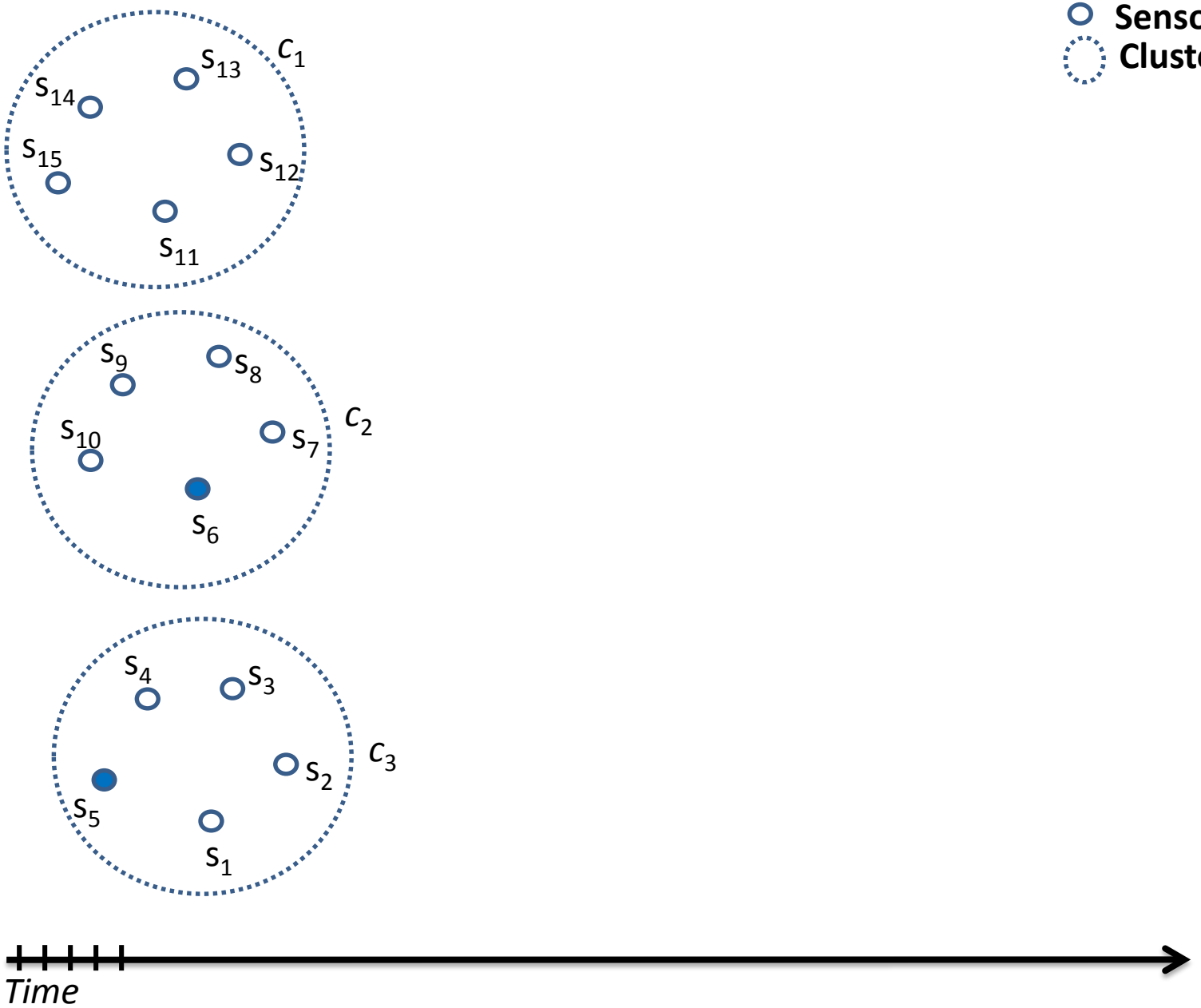
1. Snapshots may be aligned to a clock synchronized within their own network as the best approach for synchronizing numerous interacting things in real-time. That is, a global clock may be too burdensome for sensor networks that operate in the wild. Others, however, argue in favor of a global clock [Li 2004]. This article does not recommend either scheme, but acknowledges its great importance to IoT
2. Sensors release data that is either event-driven, human-driven, or released at pre-defined snapshots
3. NoTs will affect business performance – sensing, communicating, and computing can speed-up or slow-down a NoT's workflow and therefore affect the "perceived" performance of the environment it operates in or controls
4. Snapshots maybe tampered with, making it unclear when events actually occurred, not by changing *true* time (which is not possible), but changing the snapshot at which an event in the workflow triggers, e.g., sticking in a **delay()** function call


s_{14}
s_{13}
s_{15}
s_{12}
s_{11}
s_9
s_8
s_{10}
s_7
s_6
s_4
s_3
s_5
s_2
s_1
Time

Cluster

Third Primitive: Cluster – a grouping of sensors that can appear and disappear instantaneously. The basic properties and assumptions about clusters are:

1. Clusters are abstractions of a set of sensors or a network of sensors - clusters that may be created in an *ad hoc* manner versus being organized according to fixed rules
2. Clusters are not inherently physical
3. C_i is a *cluster* of n ≥ 1 sensors, $\{s_1, s_2, s_3, ..., s_n\}$
4. C_i may share one or more sensors with C_k, where $i \neq k$
5. *Late or continuously-binding* of a sensor to a cluster may result in little ability to mitigate trustworthiness concerns
6. Clusters can change their collection of sensors over time


Sensor
Cluster
c_1
s_{13}
s_{14}
s_{15}
s_{12}
s_{11}
s_9
s_8
c_2
s_{10}
s_7
s_6
s_4
s_3
c_3
s_2
s_5
s_1
Time

Aggregator

Fourth Primitive: Aggregator – a software implementation based on mathematical function(s) that transforms various sensor data into *intermediate* data.

The basic properties and assumptions about aggregator are:

1. Aggregators are virtual
2. An aggregator is assumed to lack computing horse-power, however this assumption can be relaxed by changing the definition and assumption of virtual to physical, e.g. firmware, microcontroller or microprocessor. The aggregator will use weights (See next primitive) to compute intermediate data
3. For each virtual cluster there should be a aggregator or set of potential aggregators from which to chose
4. There may be groupings of aggregators that are somehow related, e.g., with slight differences such as different weights
5. Aggregators may be acquired off-the-shelf
6. Security is a concern for aggregators (malware or general defects)
7. Reliability is a concern for aggregators (general defects).

Weight

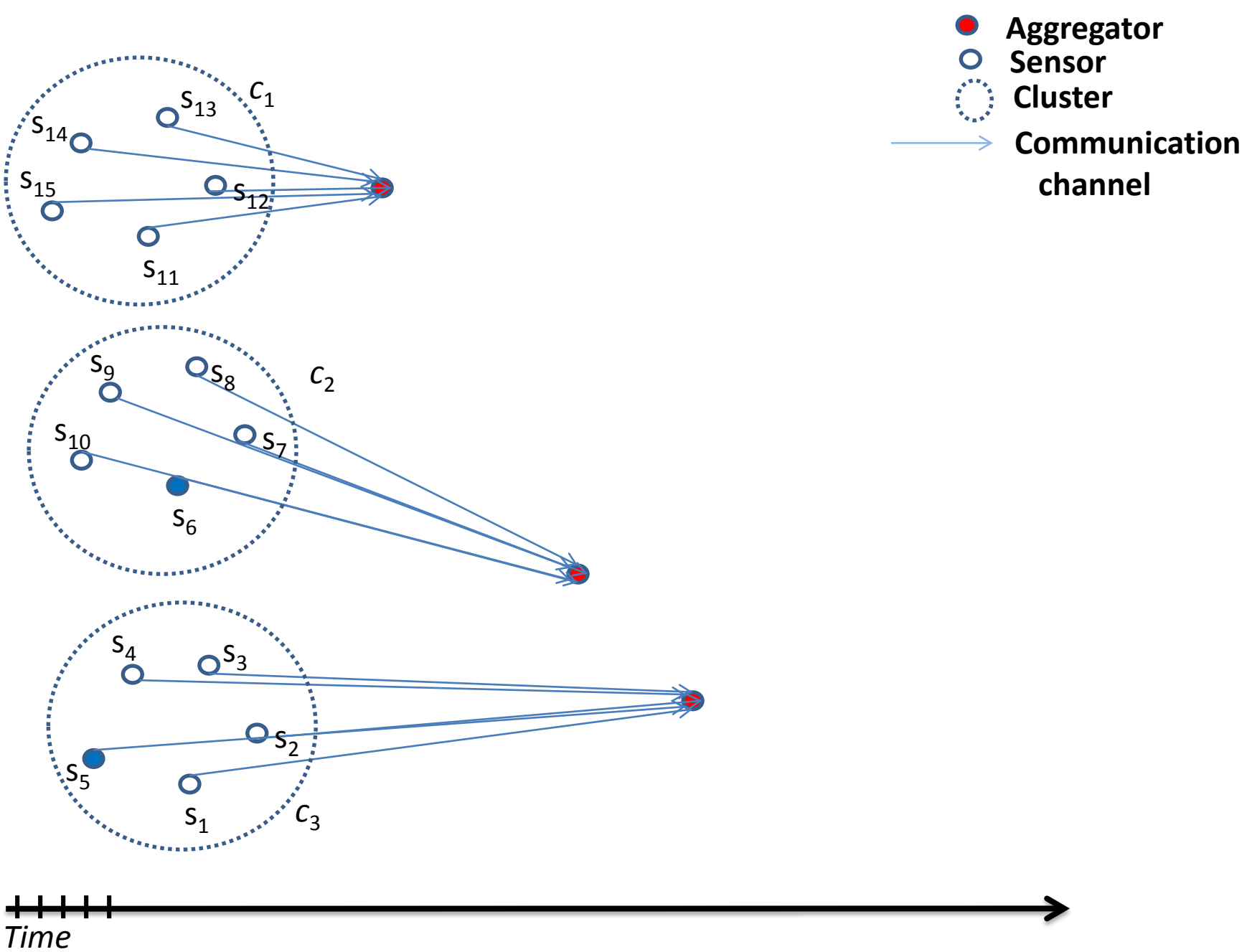
Fifth Primitive: Weight – the degree to which a particular sensor's data will impact an aggregator's computation. The basic properties and assumptions about weight are:

1. A weight can be hardwired or modified on the fly
2. A weight can be based on a sensor's perceived trustworthiness, e.g., based on who is the sensor's owner, manufacturer, geographic location of manufacture, geographic location where the sensor is operating, sensor age or version, previous failures or partial failures of sensor, sensor tampering, sensor delays in returning data, etc. A weight could also be based on the value of the data, uniqueness, relation to mission goals, etc.
3. Different NoTs can leverage the same sensor data and re-calibrate the weights per the goal of the individual NoT
4. There may be groupings of aggregators that are somehow related, e.g., with slight differences such as different weights
5. Aggregators may have intelligence and the ability to self-modify their abstract clusters as well as to modify weights

Communication Channel

Sixth Primitive: Communication Channel – any medium by which data is transmitted (e.g., physical via USB, wireless, wired, verbal, etc.). The basic properties and assumptions about communication channel are:

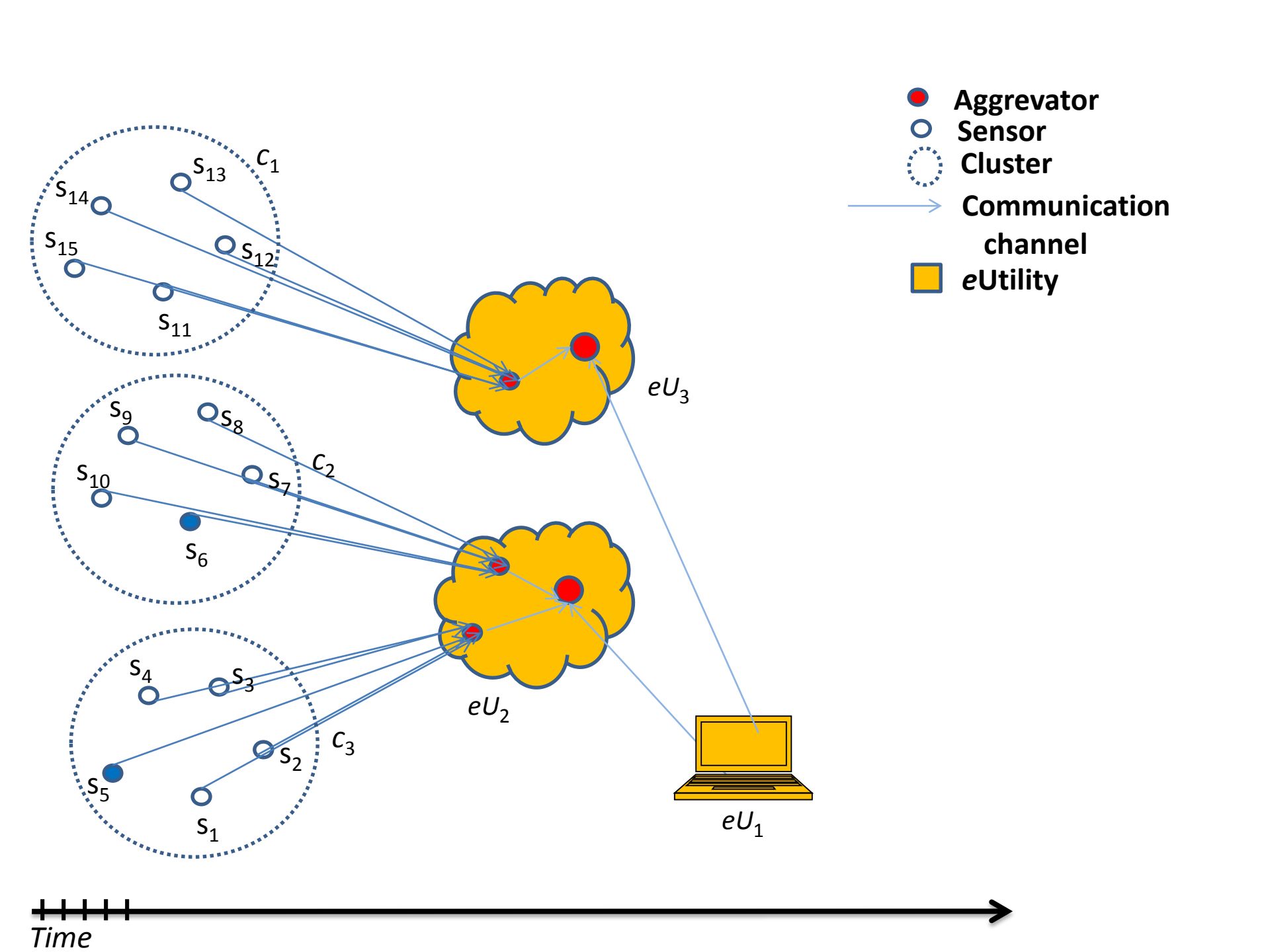
1. Communication channels move data between computing and sensing
2. Communication channels are shown as unidirectional in this primitives model, a reasonable assumption when the sensors are dumb. But the communication channel will not always be unidirectional. There are a number of conditions where an aggregator might query more advanced sensors, or potentially recalibrate them in some way (e.g., request more observations per time period)
3. Communication channels are often wireless
4. Communication channels are likely an offering (*service* or *product*) from 3rd party vendors
5. Communication channel *trustworthiness* affects the ability to move data and may make sensors appear to be failing when actually the communication channel is failing.
6. Communication channels are prone to disturbances, interruptions, and reduced reliability
7. *Redundancy* can improve communication channel reliability
8. Security is a concern for communication channels
9. Reliability is a concern for communication channels
10. Performance and availability of communication channels will greatly impact any NoT that has time-to-decision requirements (the Decision trigger primitive is discussed later).


Aggregator
Sensor
Cluster
Communication channel
c_1
s_{13}
s_{14}
s_{15}
s_{12}
s_{11}
c_2
s_9
s_8
s_{10}
s_7
s_6
s_4
s_3
s_2
s_5
s_1
c_3
Time

*e*Utility

Seventh Primitive: *e*Utility (external utility) - a software or hardware product, or service, that executes processes or feeds data into the overall dataflow of the NoT. The basic properties and assumptions about *e*Utility are:

1. *e*Utilities will likely be acquired off-the-shelf
2. *e*Utilities could be databases, mobile devices, misc. software or hardware systems, clouds, computers, CPUs, actuators, etc. Note that the *e*Utility primitive can easily be broken into different type classes
3. *e*Utilities such as clouds will provide the compute power that aggregators will likely not have
4. A human can be treated as an *e*Utility
5. Data supplied by an *e*Utility can be weighted
6. Any *e*Utility could be counterfeit (This is covered later by the Device_ID element)
7. Non-human *e*Utilities may or may not have Device_IDs; Device_IDs will likely be crucial in any authentication issues that occur for a NoT
8. Security is a concern for *e*Utilities
9. Reliability is a concern for *e*Utilities.


Aggrevator
Sensor
Cluster
Communication channel
eUtility
c_1
s_{13}
s_{14}
s_{12}
s_{15}
s_{11}
eU_3
s_9
s_8
c_2
s_{10}
s_7
s_6
eU_2
s_4
s_3
c_3
s_2
s_5
s_1
eU_1
Time

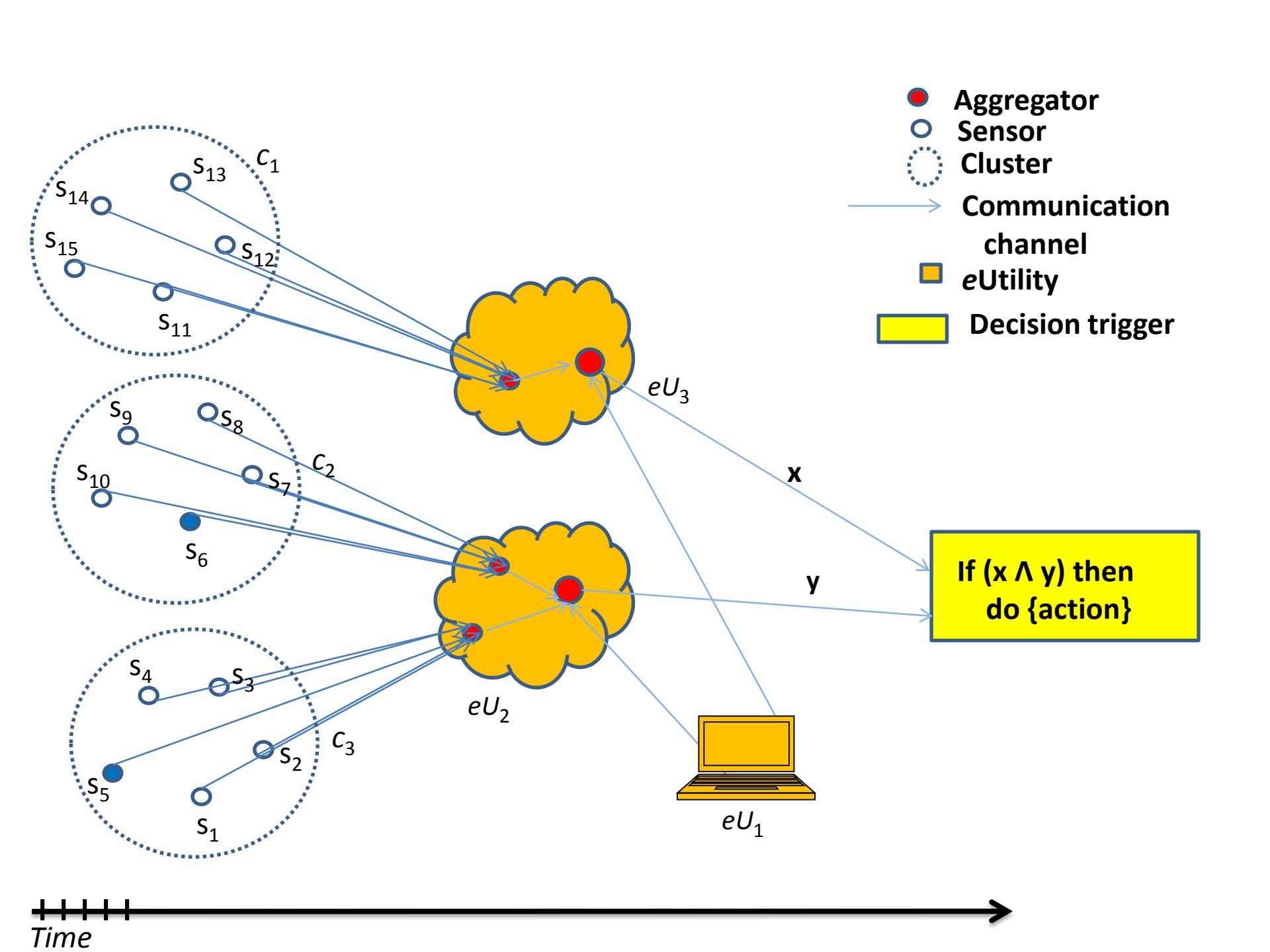
Decision trigger

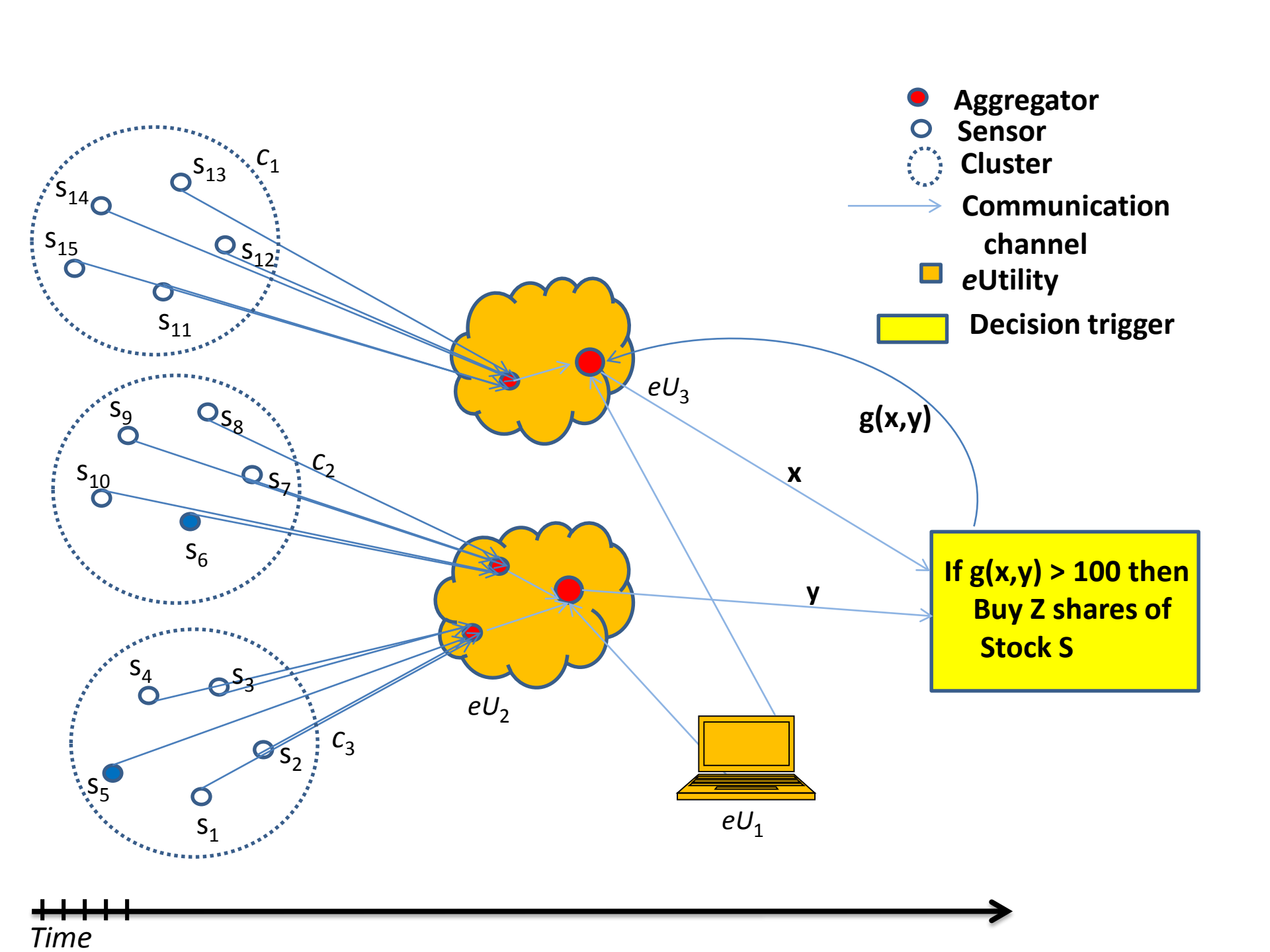
- **Eighth Primitive: Decision trigger** - the final executor of data concentrations and any other data needed to satisfy the purpose and requirements of a specific NoT. D = *f*(**x**, **y**), as shown in Figure 5, determines whether a particular action is taken. D = *f*(**x**, **y**) is the end-purpose of that NoT. The basic properties and assumptions about decision triggers are:

1. A decision trigger has an unique owner
2. Decision triggers may be acquired off-the-shelf or homegrown – homegrown seems more likely
3. Decision triggers are executed at a time snapshot and may occur continuously as new data becomes available
4. Decision trigger results may be predictions, such as whether a stock is likely to increase or decrease in value
5. Decision trigger results may control actuators or other transactions
6. If a decision trigger feeds data signals into an actuator, then the actuator should be considered as an *e*Utility if the actuator feeds data back into the NoT
7. It is fair to think of a decision trigger as an **if-then** rule, though they will not all have this form
8. The workflow from sensor data collection to decision execution may be partially parallelizable
9. Failure to make accurate decisions at time snapshot t_x may result from tardy data collection, inhibited sensors or *e*Utilities, inhibited communication channels, slow aggregators, and a variety of other subsystem failure modes

Decision trigger (cont'd)

10. Economics will play an important role in setting the rules for all upstream sensor data processing in the workflow and other processing that eventually feeds into a final decision, although workflows may operate in a continuous loop versus in batch mode
11. There may be intermediate decisions at any point in the NoT workflow before a final decision(s) results
12. Decision triggers act similarly to aggregators, and could be thought of a special case of aggregator
13. Security is a concern for decisions (malware or general defects)
14. Reliability is a concern for decision triggers (general defects). Decision triggerss could be inconsistent, self-contradictory, and incomplete. Understanding how bad data propagates to affects decisions is paramount.


Aggregator
Sensor
Cluster
Communication channel
eUtility
Decision trigger
c_1
s_{13}
s_{14}
s_{12}
s_{15}
s_{11}
eU_3
s_9
s_8
c_2
s_{10}
s_7
s_6
x
y
If (x Λ y) then do {action}
s_4
s_3
eU_2
c_3
s_2
s_5
s_1
eU_1
Time


Aggregator
Sensor
Cluster
Communication channel
eUtility
Decision trigger
c_1
s_{13}
s_{14}
s_{12}
s_{15}
s_{11}
eU_3
g(x,y)
x
s_9
s_8
c_2
s_{10}
s_7
s_6
If g(x,y) > 100 then
Buy Z shares of
Stock S
y
s_4
s_3
eU_2
c_3
s_2
s_5
s_1
eU_1
Time

The Foundation: Six Other Elements

1. **Data** – The flow of information in a NoT's workflow; data may be transferred virtually or by physical means.
2. **Environment** – The universe that all primitives in a private NoT operate in; this is essentially the *operational profile* of a private NoT. An analogy is the various weather profiles that an aircraft operates in or a particular factory setting that a NoT operates in. This may be very difficult to correctly define.
3. **Cost** – The expenses, in terms of time and money, that a specific private NoT incurs in terms of the non-mitigated reliability and security risks, as well as the costs associated with each of the primitives needed to build the private NoT. Cost is an estimation or prediction.
4. **Geographic location –** Physical place where a sensor or *e*Utility operates or was manufactured. Manufacturing location is a supply chain trust issue. Note that the operating location may change over time. Note that a sensor's or *e*Utility's geographic location along with communication channel reliability may affect the ability to move data throughout the workflow in a timely manner. Geographic location determination may sometimes be not possible.
5. **Owner** - Person or Organization that owns a particular sensor, communication channel, aggregator, decision trigger, or *e*Utility. There can be multiple owners for any of these five. Note that owners may have nefarious intentions that affect overall trust. Note further that owners may remain anonymous.
6. **Device_ID** – A unique identifier for a particular sensor, communication channel, aggregator, decision trigger, or *e*Utility. This will typically originate from the originator of the entity, but it could be modified or forged.

The Goal: Composition and *Trust*

Primitive or Actor	Attribute	Pedigree an Issue?	Reliability an Issue?	Security an Issue?
Sensor	Physical	Y	Y	Y
Snapshot (time)	Natural phenomenon	N/A	Y	?
Cluster	Abstraction	N/A	?	?
Aggregator	Virtual	Y	Y	Y
Weight	Variable constant	N/A	Y	?
Communication channel	Virtual or Physical	Y	Y	Y
eUtility	Virtual or Physical	Y	Y	Y
Decision trigger	Virtual	Y	Y	Y
Geographic location	Physical (possibly unknown)	N/A	?	?
Owner	Physical (possibly unknown)	?	N/A	?
Data	Virtual	Y	Y	Y
Environment	Virtual or Physical (possibly unknown)	N/A	Y	Y
Cost	Partially known	N/A	?	?
Device_ID	Virtual	Y	?	Y

Summary

1. A common vocabulary is useful to foster dialogue concerning IoT
2. 8 primitives that impact the trustworthiness of NoTs are proposed
3. 6 elements that impact the trustworthiness of NoTs are proposed
4. These 14 suggests a way to *define* IoT *behaviors*
5. It is unlikely a single, usable definition of IoT can be created and agreed upon
6. NoTs are the likely means by which IoT will be delivered
7. IoT is, in part, a *big data* problem (maybe "overwhelming" is more accurate than "big")
8. The goal is to someday build definitions of IoT, and better address this assertion:

   ***Trust* in some NoT *A,* at some snapshot *X,* is a function of NoT *A*'s assets ϵ {sensor(s), cluster(s), aggregator(s), weight(s), communication channel(s), *e*Utility(s), decision trigger(s)} with respect to the members ϵ {geographic location, owner, data, environment, cost, Device_IDs}, for each asset in the first set, when applicable.**

Further Reading

A publication with all of this information is available free of charge from:

http://dslsrv.gmu.edu/The%20Foundations%20of%20IoT_v2.5.pdf

Appendix: Additional Points to Ponder

Using our previous terminology, it might be useful to mention which primitives would be considered as 'things.' Those are: Sensor, Communication channel, *e*Utility, Aggregator, and Decision trigger. Note that not all primitives are 'things.'

1. Things may be all software or hardware, a combination, or human.
2. Things may have a stealth/invisible mode when coming and going thus creating near-zero *traceability*.
3. Threats to previous genres of distributed, networked systems apply to NoTs. Security threats in NoTs may be exacerbated as a result of composing seemingly limitless numbers of 3rd party things. This may create *emergent* classes of new threats.
4. Successful functional composition of things does not suggest the secure composition of the same things.
5. Forensics concerning security, for seemingly limitless numbers of continuously-binding heterogeneous things, is unrealistic.
6. 'Counterfeit things' is a *supply-chain* problem, even for software.
7. *Authentication* addresses the 'Who's Who' and 'What's What' questions. Things may misidentify, for faulty or nefarious reasons.
8. *Actuators* are things; if fed malicious data from 'other things', issues with life-threatening consequences are possible.
9. The workflow in NoTs is *time*-sensitive. Defective local or semi-global clocks (timing failures) can lead to deadlock, race conditions, and other classes of system-wide NoT failures.
10. Some NoTs may have the ability to self-organize and self-modify (self-repair). If true, NoTs can potentially rewire their security policy mechanisms and implementations or disengage them altogether.
11. *Fault injection* is a simple yet effective way to test 'things' and how their data anomalies propagate within an IoT.
12. *Fault seeding*, the inclusion of deliberately flawed sensor data, is one way of assessing integrity, via fault injection.