Is There a Science Behind the Internet of Things?

(White paper, Version 4.3)

J. Voas (j.voas@ieee.org)

Abstract

Ten core primitives belonging to most distributed computing systems, and in particular, systems with large amounts of data, scalability concerns, heterogeneity concerns, temporal concerns, actors of unknown pedigree and possible nefarious intent, is presented. Primitives allow formalisms, reasoning, simulations, and reliability and security risk-tradeoffs to be formulated and argued. These 10 primitives are basic building blocks for a Network of ‘Things’ (NoT), including the Internet of Things (IoT), an emerging ‘new’ distributed computing paradigm. They are:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Sensor</td>
</tr>
<tr>
<td>2.</td>
<td>Snapshot (time)</td>
</tr>
<tr>
<td>3.</td>
<td>Cluster</td>
</tr>
<tr>
<td>4.</td>
<td>Aggregator</td>
</tr>
<tr>
<td>5.</td>
<td>Weight</td>
</tr>
<tr>
<td>6.</td>
<td>Communication channel</td>
</tr>
<tr>
<td>7.</td>
<td>eUtility</td>
</tr>
<tr>
<td>8.</td>
<td>Decision</td>
</tr>
<tr>
<td>9.</td>
<td>Geographic location</td>
</tr>
<tr>
<td>10.</td>
<td>Owner</td>
</tr>
</tbody>
</table>

A composability model and vocabulary that defines principles common to most, if not all NoTs, is needed. For example, “what is the science, if any, underlying the IoT”? Primitives offer answers by allowing comparisons between one NoT architecture to another. They offer a unifying vocabulary that allows for composition and information exchange among differently purposed networks. And they prove useful towards more subtle concerns, including interoperability, composability, and late-binding of assets that come and go on-the-fly, all of which are large concerns for IoT.

Introduction

After surveying published IoT literature and products, it is clear there is value in having numerous sensory devices connected to a larger infrastructure. ‘Use cases’ already exist that illustrate that such a paradigm is valuable and that this paradigm will be the next evolutionary step for technology. However, the current IoT landscape presents itself as a confusing mix of buzzwords, consumer products (e.g., smart homes), and over sensationalized predictions. There is no clear formal, analytic, or even descriptive set of primitives that govern the operation, security, and lifecycle of the IoT. There is a
vacuum between the hype and the science, if a science exists.

For clarity, note that this paper uses the terms IoT and NoT interchangeably - the relationship between NoTs and IoT is simple, yet subtle. IoT is an instantiation of a network of things, and in particular, IoT has its ‘things’ tied 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, industrial internet, and cyber-physical systems 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 will prove invaluable as there is no one static IoT, contrary to current discourse.

The Model

This paper is based on primitives. Each primitive, assumptions about that primitive, and its role is now discussed. This paper uses a data-flow model captured as a sequence of 5 figures to explain how primitives compose with each other and affect confidence that a particular NoT is trustworthy.

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

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. Security is a concern for sensors if they or their data is tampered with or stolen

19. Reliability is a concern for sensors.

**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 assumptions about snapshot are:

1. Sensors release data that is either event-driven, human-driven, or released at pre-defined snapshot times

2. 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

3. NoTs may affect business performance – sensing, communicating, and computing can speed-up or slow-down a NoT’s workflow and therefore affect the performance of the environment it operates in or controls

4. Snapshots maybe tampered with making it unclear when events actually occurred.
In Figure 2, 15 sensors are shown – the blue sensors indicate sensors that are ‘somehow’ failing at specific snapshots, that is, they are not satisfying their purpose and expectations. As mentioned earlier, there could be a variety of sensor failure modes, some temporal, and some relate to data quality. Further the temporal failure modes for sensors may be actually a result of the means of transport of that data failing, and not the sensors. More on point that later.

(A good reference to the research issues related to snapshots is found in [NIST 2015].)

Figure 2: Snapshots
**Third Primitive: Cluster** – a grouping of sensors that can appear and disappear instantaneously. The assumptions about clusters are:

1. Clusters are abstractions of a set of sensors or a network of sensors that may or may not be *ad hoc*
2. Clusters are not inherently physical
3. $C_i$ is a *cluster* of $n \geq 2$ sensors, \{s$_{1}$, s$_{2}$, s$_{3}$, ……, s$_{n}$\}
4. $C_i$ may share a sensor with $C_k$, where $i \neq k$
5. *Late-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

Figure 3 shows 3 clusters with 5 sensors assigned to each, however note assumption 6 above.
**Fourth Primitive: Aggregator** – is a software implementation based on mathematical function(s) that transforms various sensor data into *intermediate* data. The associated assumptions are:

1. Aggregators are virtual
2. An aggregator is assumed to lack computing horse-power, however this assumption can be relaxed by changing their definition from 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).

**Fifth Primitives: Weight** – is the degree to which a particular sensor’s data will impact an aggregator’s computation with associated assumptions:

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.
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

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

1. Communication channels move data between computing and sensing
2. Communication channels are shown as unidirectional in this simple 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 primitive is discussed later).

Figure 4 shows the data coming from each of the three clusters being input to three corresponding aggregators. It is now the responsibility of the aggregators to turn those 5 sensor inputs into three new data points. Note that there is a close relationship between a cluster and an aggregator. For example, in Figure 4, aggregator $C_1$ might be determining how busy restaurant $A$ is. Five independent sensors in $A$ could be taking pictures from inside and outside (parking lot) of $A$, room temperature measurement in the kitchen, motion detectors from the dining area, sound and volume sensors, light detectors, etc. So while the sensors are certainly not homogeneous, their data is processed to make a new piece of data to address one question with possible results such as very busy, not busy, closed, etc. And aggregators $C_2$ and $C_3$ might be doing the same for restaurants $B$ and $C$ respectively.
Seventh Primitive: eUtility (external utility) - a software or hardware product, or service, that executes processes or feeds data into the overall dataflow of the NoT. The assumptions about eUtilities are:

1. eUtilities will likely be acquired off-the-shelf
2. eUtilities could databases, mobile devices, misc. software or hardware systems, clouds, computers, CPUs, etc. Note that the eUtility primitive can easily be broken into different type classes
3. eUtilities such as clouds will provide the compute power that aggregators will likely not have
4. A human can be treated as an eUtility
5. Data supplied by an eUtility can be weighted
6. Any eUtility could be counterfeit
7. Security is a concern for eUtilities.

8. Reliability is a concern for eUtilities.

Figure 8 illustrates the use of two cloud eUtilities executing the functions of five aggregators. (These different clouds might be from different vendors.) Figure 8 shows the addition of one non-cloud eUtility, eU₁ (a laptop), in this NoT.

**Figure 8: eUtility**

---

**Eighth Primitive: Decision** - A decision is the final result from data concentrations and any other data needed to satisfy the purpose and requirements of the specific NoT. \( D = f(x, y) \), as shown in Figure 9, determines whether a particular action is taken. \( D = f(x, y) \) is the end-purpose of that NoT. Going back to our restaurant example, if \( C_2 \) did something similar for restaurant \( B \) and \( C_3 \) for restaurant \( C \), and the laptop sent in data concerning the calendar and times when \( A \), \( B \), and \( C \) were open, then variables \( x \) and \( y \) in Figure 9 might be a data point as to whether restaurants had customers during their daily open-for-business times. And obviously \( x \) and \( y \) could be refreshed for as many snapshots as desired. The output of the decision might be valuable information for a competing restaurant or a corporation if \( A \), \( B \), and \( C \) were parts of a restaurant brand.
Decisions are the outputs of NoTs and the reason for the existence of NoTs. The assumptions about decisions are:

1. A decision has an unique owner
2. Decisions may be acquired off-the-shelf or homegrown – homegrown seems more likely
3. Decisions are made at a time snapshot and may occur continuously as new data becomes available
4. Decisions may be predictions, such as whether a stock is likely to increase or decrease in value
5. Decision results may control actuators or other transactions
6. It is fair to think of a decision as an if-then rule
7. The workflow from sensor data collection to decision making is partially parallelizable
8. Failure to make accurate decisions at time snapshot \( t_s \) may result from tardy data collection, inhibited sensors or eUtilities, inhibited communication channels, slow aggregators, and a variety of other subsystem failure modes
9. 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
10. There may be intermediate decisions at any point in the network of things workflow before a final decision(s) results
11. Decisions act similarly to aggregators, and could be thought of a special case of aggregator
12. Security is a concern for decisions (malware or general defects)
13. Reliability is a concern for decisions (general defects). Decisions could be inconsistent, self-contradictory, and incomplete. Understanding how bad data propagates to affects decisions is paramount.
Figure 9: Decision

If \((x \land y)\) then do \{action\}
**Ninth Primitive: Geographic Location** - definitions (1) and (2) and one assumption (3) are:

1. Place where sensor or eUtility operates – these may change over time
2. Place where sensor or eUtility was manufactured
3. A sensor’s or eUtility’s geographic location along with communication channel reliability may affect the ability to move data throughout the workflow in a timely manner

**Tenth Primitive: Owner** - definition (1) and one assumption (2) are:

1. Person or Organization that owns a particular sensor, communication channel, aggregator, decision, eUtility, or computing platform
2. There can be multiple owners for any entity in a NoT
3. Owners may have nefarious intentions.

**Three Other Actors**
To complete our model, there are three other actors, *data, environment, and cost*, that although not primitives, are driving forces in terms of how private NoTs will be architected. These additional actors play a major role in fostering the degree of trustworthiness\(^1\) that a specific private NoT can provide.

1. **Data** – the flow of information in a NoT workflow; data may be virtual or physical,
2. **Environment** – the universe that all primitives in a private NoT operate in; this is essentially the 
   *operational profile* of the private NoT, and
3. **Cost** – the expenses, in terms of time and money, that any specific private NoT architecture incurs in terms of the non-mitigated reliability and security risks, as well as the costs of each of the actors and the architecting of the private NoT.

**Other Considerations**
Five other items as ‘food for thought’ include:

1. **Open-Loop, Closed-Loop**

NoTs can be open-loop, closed-loop, or something in between. So for example, an automobile can have hundreds of sensors, numerous CPUs, databases for data such as maps, wired communication channels throughout the car, and no wireless access to or from any ‘thing’ in the car to outside the car. This would be a closed-loop NoT. Clearly such a NoT mitigates wireless security concerns such as remotely controlling a car, however there could still be concerns of malware, counterfeit ‘things’ with greatly reduced reliability that could result in negative safety situations, i.e., accidents. A fully open-loop system

\(^1\) *Trustworthiness* includes attributes such as security, privacy, reliability, safety, availability, and performance, to name a few.
would essentially be any ‘thing’ talking to any ‘thing’, anyway, and at any time. This, from an architectural “trustworthiness” standpoint creates chaos. Thus most practical NoTs will be somewhere between these extremes. The primitives enumerated in this paper can serve, if just by this simple automobile example, as a guidepost as to where reliability and security concerns require mitigation or additional design and testing considerations.

2. Patterns

One could envision that there are design patterns for building NoTs from smaller NoTs, i.e., something similar to design patterns in object-oriented programming. In essence, these would be sub-NoTs. By having sub-NoTs, there would be a means of commercializing sub-NoTs as blueprints or as actual physical packages. This could speed-up adoption of NoTs for smaller organizations that are seeking to leverage this distributed computing paradigm but do not know where to begin, i.e., a catalogue of sub-NoTs.

3. Composition and Trust

To better understand the inescapable trust issues associated with IoT, it is necessary to consider the general attributes of the primitives and actors. After all, they will form a particular NoT, and the trust of that NoT needs some form of measurement.

The following table shows the diversity of these 13, and begs questions such as: what does trust mean for a NoT when its abstractions are in continual flux due to a natural phenomenon that is in continuous change and while all of its virtual and physical systems are either unknown or partially unknown or faulty? Or if you have insecure physical systems operating using faulty snapshots composed with incorrect assumed environments, what is your level of trust?

<table>
<thead>
<tr>
<th>Primitive or Actor</th>
<th>Attribute</th>
<th>Pedigree an Issue?</th>
<th>Reliability an Issue?</th>
<th>Security an Issue?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor</td>
<td>Physical</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Snapshot (time)</td>
<td>Natural phenomenon</td>
<td>N/A</td>
<td>Y</td>
<td>N/A</td>
</tr>
<tr>
<td>Cluster</td>
<td>Abstraction</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Aggregator</td>
<td>Virtual</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Weight</td>
<td>Variable constant</td>
<td>N/A</td>
<td>Y</td>
<td>N/A</td>
</tr>
<tr>
<td>Communication channel</td>
<td>Virtual or Physical</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>eUtility</td>
<td>Virtual or Physical</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Decision</td>
<td>Virtual</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Geographic location</td>
<td>Physical (possibly unknown)</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Owner</td>
<td>Physical (possibly unknown)</td>
<td>N/A</td>
<td>N/A</td>
<td>Y</td>
</tr>
<tr>
<td>Data</td>
<td>Virtual</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Environment</td>
<td>Virtual or Physical</td>
<td>N/A</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Cost</td>
<td>(possibly unknown)</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
</tbody>
</table>

Such questions are not easily answered or desirable, but they demonstrate the difficulty with the issue of trust.

### 4. IoT Definition

One increasingly louder cry in both the applied engineering and computer science communities is for a definition(s) as to what IoT is and is not. While this ‘foundation building’ effort has not reached a point where that can do that with any certainty, this effort has reached a point of understanding the composability issues related to building IoT primitives into applied engineering systems, e.g., a factory or automobile. With further hands-on experimentation, closing in on a definition(s) will continue.

Until then, the following should hold:

*Trust in NoT* $A$, at snapshot $X$, is a function of the NoT assets $e \{\text{sensor(s)}, \text{cluster(s)}, \text{aggregator(s)}, \text{weight(s)}, \text{communication channel(s), eUtility(s)}, \text{decision(s)}\}$ given that the members $e \{\text{geographic location, owner, data, environment, cost}\}$ are considered at snapshot $X$ for each asset in the first set, when applicable.

### 5. Fault Types

In a computing infrastructure where terms like unknown, anything, and unexpected cannot be wished away, the same terms will hold for the fault and failure modes that will be prevalent. While this effort has had no experience on the topic yet, this effort expects to begin looking at various well-established fault and failure taxonomies to predict which might apply to IoT such as [Wallace 2001].

### Summary

This white paper offers a glimpse into the main primitives of most distributed computing systems. It has attempted to organize and present the primitives in a way that more appropriately mirrors the emerging ‘networks of things’ paradigm. To do so, a common vocabulary has been proposed that is simple while useful enough to foster dialogue concerning the IoT, one type of a NoT. Ten primitives which impact the trustworthiness of NoTs are proposed, and three other actors address a private NoT’s data, environment, and costs.

By having primitives, analytics and formal arguments about “use case” scenarios address the “what if” questions and scenarios that most vendors and current IoT practitioners avoid. These scenarios can yield a quick determination as to which existing security approaches apply, to what degree, and at what cost. For example, authentication can be used in addressing issues such as geo-location and ownership of sensors but maybe that is not relevant if an adversary “owns” one of the sensors and can obtain that information based on proximity. Encryption can protect sensor data transmission integrity and confidentiality including cloud-to-cloud communication but it might render the IoT sensors unusable due
to excessive use of energy. While fault-tolerance techniques can alleviate reliability concerns related to defective 3rd party ‘things’ that are often inexpensive and replaceable, they can be insecure and induce communication overhead and increased attack surfaces. So in short, primitives and how they can be composed create a roadmap for how to apply existing technologies that support confidence in IoT trustworthiness. And this simple “primitives” model can become the design blueprint that forewarns when certain architectures of hybrid NoTs and IoT systems will be too slow to enforce concepts such as net neutrality. In short, our primitives are really just objects with attributes, with the 10 forming a design catalog.

This effort acknowledges that there is likely better labeling for these actors and primitives, and that even a reduction or increase in the number of them, depending on perspective, could prove beneficial. This is a first attempt to grasp at a possible underlying science. For example, one could easily argue that snapshot, geographic location, and owner are not primitives. While semantically true, the aforementioned trust statement needs refinement and must incorporate them as variables or constants.

It is reasonable to believe that private NoTs (or ‘subnets of things’) are the likely means by which IoT will be delivered as a practical commodity, such as in applied engineering. Further, NoTs will fall victim to the big data quagmire, since as the number of clusters expands or explodes, so does the amount of data.

IoT-induced public perception cannot be overlooked, and the control of data and when events trigger are noteworthy weapons available to those with nefarious intent. If one can tamper with event sequencing and data content, commercial decisions, elections, and public trust can be undermined.

And finally, most agree that standards are needed at the IoT implementation level [Machina 2014]. This white paper is aimed to be guidance for SDOs to define such standards.

So is there a science behind the IoT? Or is it engineering? Or is it a new bottle with old wine representing the past 40 years of distributed computing practices? That’s not clear, but what is is that there is a science out there somewhere for the ‘things’ and composability formulae needed to make this happen on a trustworthy, grand scale.

A recent comment at a conference sums it up nicely “We’re in a physical world at the speed of software.”

Hopefully this article has shed light, and sparked curiosity.

**Future Work**

Three diverse use case studies are being architected: (1) acute healthcare (high security, reliability, and safety concerns), (2) smart home (high security, reliability, and safety concerns), and (3) crop agriculture (low security, medium reliability, and little to no safety). The benefit of creating plausible use cases is that it creates “vanilla” NoT molds/patterns that could apply to other named NoTs: Cyber Physical
Systems (CPS), sensor networks, Industrial Internet (II), social media networks, etc. Eventually a set of use and misuse cases for a variety of these domain themes will be developed.

Acknowledgement
Many thanks to Joseph Williams (Seattle Pacific University), Dave Ferraiolo (NIST), Tim Hall (NIST), Rick Kuhn (NIST), and Phil Laplante (Penn State U.) for comments on earlier drafts. And thanks to Angelos Stavrou (George Mason U.) for encouraging this line of thinking.

References


About the Author
Jeffrey Voas is a computer scientist at the US National Institute of Standards and Technology (NIST) in Gaithersburg, MD. Voas is a Fellow of the Institute of Electrical and Electronics Engineers (IEEE), the Institution of Engineering and Technology (IET), and the American Association for the Advancement of Science (AAAS).
Appendix: Additional Points to Ponder

1. Things may be all software or hardware, a combination, or human. [system or systems]

2. Things may have a stealth/invisible mode when coming and going thus creating 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.


5. Forensics concerning security for seemingly limitless numbers of late-binding heterogeneous things is unrealistic.

6. ‘Counterfeit things’ is a supply-chain problem, even for software [Skyba].


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 [Voas 1998].

12. Fault seeding, the inclusion of deliberately flawed sensor data, is one way of assessing integrity, via fault injection.