Event Based Performance Evaluation and Analysis of Dynamic Wireless Networks

Denis Carvin, Guillaume Kremer, Philippe Owezarski, Pascal Berthou

Event Based Performance Evaluation and Analysis of Dynamic Wireless Networks

Denis Carvin[1,2a], Guillaume Kremer[1,2b], Philippe Owezarski[1,2c], Pascal Berthou[1,2b]
[1]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[2]Univ de Toulouse, [a]INSA, [b]UPS, [c]LAAS, F-31400 Toulouse, France
Email: {carvin, kremer, owe, berthou}@laas.fr

***Abstract*—This last decade has seen an increasing interest for wireless communications. With the current use of smart-phones and tablets coupled to the rise of the Internet of Things, the number of mobile terminal nodes in networks will significantly change the way we manage them. Indeed, these wireless networks are highly dynamic, especially concerning topology and traffic matrices. The fast moves of mobile devices can for instance impact the connexity of the networks, or the importance of one of the nodes in the routing graph. Therefore distributed network control, and traffic management are of increasing complexities. Given this high dynamicity, network management will need to be as autonomous as possible regarding the service they are providing to users and their resilience. In this paper, we propose a method for monitoring and assessing the quality of a dynamic mobile network. For this purpose, we introduce the concept of System Development Index (SDI). We especially provide methods and algorithms, which are based on events collection and distributed mining to analyze the evolution of this index. We illustrated it by simulation, considering different scenarii under NS3. We evaluated the evolution of the SDI, and analyzed its possible underlying reasons for one scenario and analyze error estimation for various network properties.**

I. Introduction

Nowadays, one can say that wireless mobile networks have definitely invaded our daily lives. When looking at forecasts from Cisco [1], mobile traffic should increase 7.5-times in the 4 next years. While the service quality will be of primer interest (two-thirds of this traffic will be dedicated to video), provider systems will be tricky to manage. Managing wired networks to sustain unpredictable traffic is still a complex and unsolved technical domain. The problem raised by wireless networks will be even more complex: in particular, wireless infrastructures will experience dynamic topologies depending on terminals position and needs, thus adding a level of magnitude of complexity. Indeed, complex phenomena will arise with the Internet of Things where random interactions might generate new kinds of traffic. As a result, system managers will not be able to quickly analyze the underlying reasons that affect their networks' behaviours, and so, to launch appropriate corrective measures. Thus, this task needs to be mainly delegated to systems themselves, for providing fast autonomous adaptative actions when required (when detecting events leading to network QoS decreases, or even better when predicting such QoS decreases). In this paper we provide a reference model for an observer to assess a system and then conduct an assessment-centric analysis. While this model could be applied to systems in general, we consider self-assessment and self-analysis capabilities for dynamic wireless network of collaborative nodes. More specifically, our work is focused on the service provided by the network layer. In this particular case we will have to face two majors difficulties: (1) nodes only have partial information on the network state, (2) the level of service relies on a wireless medium which is complex and not reliable by nature. Concerning self-assessment, partial information forces nodes to agree on a value based on their local information. Also, this value is time varying and the problem is from the class of distributed consensus. Concerning self-analysis, nodes will have to search and share the information they dispose to understand the law followed by the consensus value. Then, they will have to determine whether they are responsible for its evolution. This could be seen as a distributed data-mining in uncertain partial context problem, uncertainty being harshened by the medium kind. Therefore, the remainder of this paper is structured as follows. The following section describes the related work on data-mining applied to network management and distributed consensus. We introduce in section III our general assessment and analysis framework for multi-agent systems. In section IV, we instantiate this framework in the case of wireless dynamic systems and lead an off-line analysis through an example scenario. In the fifth section, a distributed assessment algorithm is provided for our network. We also evaluate it under various network conditions. We will conclude on future work in a last section.

II. Related Work

A. Knowledge Extraction and Network Analysis

Understanding and Managing wireless network is one of the operator concern. Orange Labs have shown interest on the optimal deployment of wireless substitution network [2]. On its side, AT&T Labs worked on data-mining to analyze its own wireless infrastructure [3]. Data-mining applied to networking has already been investigated, in particular by the security community which is quite fond of this angle. The approach is applied to network intrusion detection but also in traffic monitoring and anomaly detection. The main idea is to lighten network manager task by removing false positive alert. In [4] the authors mentioned that data-mining was a valuable tool but which was not about making human analysis unnecessary, specifically in the attribute choices. Then data-mining are useful to construct new rules. For example, technical results

can be found in [5] where Casas & al. demonstrated the efficiency of clustering techniques to detect traffic anomaly and construct new filtering rules without knowledge. Also, understanding cause and effect between network events is not the stronghold of security. We found in [6]–[8] analysis concerned by the understanding of network behavior. In [6] authors highlighted the sources of TCP reset anomalies. The field of wireless communication is investigated in [7] where the key characteristics of the traffic are captured on several base stations to optimize their coordination. Authors showed a significant enhancement on the downlink delay performance by clustering users in profiles. Finally authors of [8] focused on the relation that can exist between user experience and the network quality of service. The studies was lead on a set of mobile users and explained the relation between server response time, round time trip and user satisfaction. While [5], [7] have brought methods to extract information, [6], [8] have tailored their studies toward a very specific goals. These two approaches need to be linked by a common objective which is the network performance. Also, narrowing extracted information down to assessment will allow one to take up the issue of partial information. Indeed, despite the efficient work cited above, it only considers an omniscient and centralized approach because of their domain complexity. Therefore we insert our work in between, with the motivation to only extract clues on a system relatively to its assessment. The assessment needs to be shared among all the nodes, which leads us to the distributed consensus problem. Since our assessment method relies on an average over nodes, we will focus on the average consensus problem.

B. Distributed Consensus And Average Sharing

In distributed algorithms, when agents need to agree on a value, it remains on the well known consensus problem. This class of problem has been deeply studied from various angles (termination, fault tolerance, etc...). A subclass of this problem is called average consensus, where each agent i keeps a value v_i. The consensus for agents is to find the average x of the kept values. Many algorithms have been designed, this is perfectly illustrated in [9]. Every algorithm is a trade-off between time convergence, memory used by each agent and the number of exchanged messages. Among the ones that require few communications and computational resources, the most famous are : the maximum degree weight given in [10], the metropolis weight given in [11]. In these schemes, each node proceeds iteratively, computes values and sends them to its neighborhood at each step. These linear algorithms both consider that the dynamic of the average is greater than the convergence time for a connected network. The estimation $\widehat{x}_i(t)$ of x for node i at step $t+1$ is given by :

$$\widehat{x}_i(t+1) = \alpha_i.\widehat{x}_i(t) + \sum_{j \in N_i(t)} \alpha_j.\widehat{x}_j(t) \quad and \quad \widehat{x}_i(0) = v_i$$

where $N_i(t)$ is the neighborhood of i at step t and α_k depends on the algorithm:

Maximum degree weight algorithm for N agent:

$$\alpha_k = \frac{1}{n} \quad if \quad k = i \quad and \quad \alpha_k = 1 - \frac{|N_i(t)|}{n} \quad otherwise$$

Metropolis weight algorithm:

$$\alpha_k = \frac{1}{1 + max(|N_i(t)|, |N_j(t)|)} \quad if \quad k \neq i$$

$$and \quad \alpha_k = 1 - \sum_{k \neq j} \alpha_k \quad otherwise$$

Keeping in mind the introduced related work, we can now detail our assessment and analysis framework of systems.

III. An Assessment and Analysis Framework for Multi-Agent Systems

A. System Development Index and Assessment

1) A Multi-Agent Systems Model: Inspired from the Multi-Agent Systems theory, our framework considers a System as a set of agents. An agent is an entity which owns interfaces to interact with its environment and specifically with others agents. An agent also owns interfaces to observe events or interactions that occurred to him or to its neighborhood. Each agent has an utility value that represents its wellness over time. We defined utility values in the real interval $[0, 1]$, where 0 is a worst case and 1 is a best case. The utility value of an agent can also be seen as its percentage of satisfaction driven by a possibly unknown utility function. Each agent is able to store a set of observed interactions as well as its satisfaction history. The formalism we used to describe these observations is detailed in the next section.

2) System Ideality and System Development Index: When considering system assessment, our main axioms are the following:

- Systems where all agents are 100% satisfied are ideal.
- Systems where all agents are 0% satisfied are not ideal.

Therefore we can summarize the level of ideality of a N-agents system in a N-dimensional vector of utility values.

- Let S be an N-agents system
- Let $u(t)$ be the utility vector of agents in S at time t
- Let $u_i(t)$ be the utility of agent i in S at time t
- Let z $\in [0,1]^N$ such as $z_i = 0 \quad \forall\ i \in [1, N]$
- Let o $\in [0,1]^N$ such as $o_i = 1 \quad \forall\ i \in [1, N]$

A SDI for a N-agents System S is a function from $[0,1]^N$ to $[0,1]$ that satisfies both of our axioms, thus the space of SDI function for S is given by:

$$\{g : [0,1]^N \to [0,1] \mid \quad g(o) = 1\ and\ g(z) \neq 1\}$$

The system S is said ideal for a given SDI g at time t if and only if :

$$g(u(t)) = 1$$

Thus in our framework, system ideality is related to a point of view or function. The classical form of SDI is given by

$$f(s) = \sum_{i=1}^{N} \gamma_i.u_i \quad with \quad \sum_{i=1}^{N} \gamma_i = 1$$

The value of γ_i could be based on the pricing policy of a system manager who gives preferences to some users classes. For a selfish agent i, the system will be ideal as far as its own satisfaction equals 1. In the latter case $\gamma_j = 0$ for $j \neq i$ and $\gamma_i = 1$. One can also base a SDI on a distance between the satisfaction distribution and the perfect distribution where all agents are fully satisfied. The main point is that a SDI is an index that one wants to follow. For an external observer, this index can just be studied. In the case of a concerned observer like a system manager, this index will be tracked in order to be maximized. Finally if the observer is an agent, the index indicates how well the system in which it evolves is ideal given its point of view.

3) SDI Estimation and Analysis in Various Systems: Computing the value of the SDI in real time is not trivial for systems with a large number of nodes. Indeed one needs to have access in real time to the utility value of each agent to be able to compute its exact SDI. From an agent point of view, it means that he should collaborate with others, which is not always the case. For the same reasons, accessing the necessary information to understand the evolution of the SDI is not always possible. We can thus study the SDI with different angles which depend on omniscience, and interactivity (off-line vs on-line):

Omniscient Off-line Studies: This kind of study considers the whole set of information contained by the agents and analyzes it passively. The aim here is to identify behaviors, cause-and-effect regarding the evolution of the SDI for a whole system. It allows one to retrieve knowledge on the underlying reasons that drive the assessment of a system under a given point of view (SDI). Consequently, these studies are preliminary studies.

Partially informed Off-line Studies: In this case the information considered is only contained by a subset of agents (or a unique agent) under a passive analysis. The main goal of these studies is to characterize the possible conclusion that a subset of agents could draw with partial information.

On-line Studies: While for Off-line study it is possible to be omniscient, in the case of On-line studies, the information will always be considered as partial, since the agents can not know the whole state of the system. Thus they can not be sure of the consequences of their actions. That is the reason why this type of analysis better requires preliminary off-line survey and calibration.

B. Event Based SDI Analysis

In order to analyze the evolution of a system, we consider its initial state and its succession of events. We have previously described our model such as a set of agents having interfaces through which they observe events. In this section we will detail the formalization of these observations and the way to analyze them.

1) Observation and Event Definitions: An observation can be seen as tri-dimensional point. More specifically an observation has a time dimension, an observer (or agent) dimension and an event dimension where events are also multidimensional. An example of table is illustrated in table I.

Time: The time dimension is crucial since we want to study and manage the temporal evolution of the SDI. Time is considered continuous, second(s) is the principal unit. We keep only one time representation which is called *#Time*. In our implementation, we considered time as a float value with the experiment start as the origin.

Agent: Agents are the main entities of our systems they interact each others, observe, analyze and make decision. They will have a unique identifier *#Agent*.

Event: An event is a *perceptible* modification of the system state. Combining the initial state and events, one can trace a partial history of a system. As specified above, an event is a multidimensional object identified by a primary key *#Event*. It can be represented by a frame where the first field is the event type (*eType*) which determines the validity and the meaning of the following ones. An Event can occur several times and be observed at different moments by distinct agents.

(a) Example of Observations Table

Observations		
#Time	**#Agent**	**#Event**
float	*int*	*int*
1.2	0	0
1.25	1	1
1.255	0	1
1.3	1	0

(b) Example of Event Table

Events					
#Event	**eType**	**eSource**	**eSpeed**	**eLength**	...
int	*string*	*int*	*float*	*int*	...
0	'Move'	0	3.0	-	...
1	'Packet'	1	-	1500	...

TABLE I

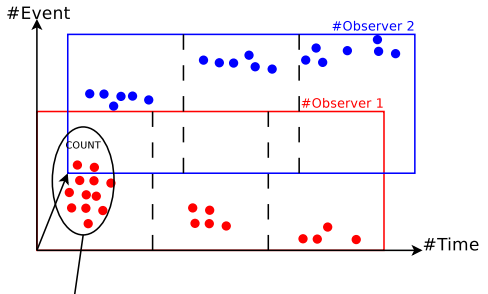
IN THIS SCENARIO, WE HAVE TWO AGENTS, EACH OF THEM PRODUCES AN EVENT. EACH EVENT IS OBSERVED BY BOTH AGENTS. AGENT 0 MOVES AT TIME 1.2S WHILE AGENT 1 SENDS A PACKET AT TIME 1.25S. AGENT 0 OBSERVES THE SAME PACKET 5MS LATER WHILE AGENT 1 REALIZES THAT AGENT 0 HAS MOVED AT 1.3S. THE TWO EVENTS HAVE A FIELD IN COMMON (*eSource*) AND DISTINCT ONES (*eLength* AND *eSpeed*)

2) Constructing Observation Features: The idea behind an event based SDI analysis is to link the evolution of an SDI to the evolution of observations features. We call feature a property of an observation cluster. Thus, we will create clusters of observations, compute some cluster properties that vary over time and then study the association between these properties and the SDI. As a result, when proceeding to an SDI analysis, one wants to define three important things : (1) An algorithm to define clusters, (2) distance functions between observations and (3) the properties to observe. Therefore we give in table II examples of canonical distances that one could use to build a distance between observations.

Dimension	Distance(o1,o2)
Time	abs(o1.#time-o2.#time) abs(HoD(o1.#time)-HoD(o2.#time)) abs(DoW(o1.#time)-DoW(o2.#time))
Agent	2-norm(o1.#agent.position,o2.#agent.position) o1.#agent.nbHop(o2)+o2.#agent.nbHop(o1)) RTT(o1.#agent,o2.#agent)
Event	LevenshteinDist(o1.#Event.eType,o2.#Event.eType) card({field | o1.#Event.field≠'-' ⊕ o2.#Event.field≠'-'})
Generalized	Same(f(o1),f(o2)) where Same(x,y)=0 if x=y ; ∞ otherwise example : f(x)=(x.#agent,seconds(x.#time))

TABLE II
THIS TABLE GIVES CANONICAL DISTANCES FOR EACH DIMENSION TO BE USED WHEN GROUPING OBSERVATIONS. FOR TIME DIMENSION WE SUGGEST THE USE OF HOUR OF DAY (HOD) OR DAY OF WEEK(DOW). FOR EVENT DIMENSION, THE STRING COMPARISON OF TYPE NAME IS POSSIBLE (LEVENSHTEIN OR EDIT DISTANCE). IN THIS PAPER, WE WILL USE A GENERALIZED DISTANCE (SAME) BASED ON LAMBDA FUNCTIONS (IN OUR CASE KEY COMPARISON)

Regarding time, one can express naturally the distance between two timestamps as a simple difference of the values in seconds. Nevertheless, depending on the studied system, it could be meaningful to use seasonal distance like the difference between hour of day or day of week. When considering agents, the natural way to evaluate distance is to use geographical positions. This last approach might not make sense if agents are software entities in a same physical system. Then, analysts might want to define other distances like the proximity of their state or the number of hops in the case of networks. Distance between events are less obvious to determine. However it is still possible to create generic metrics based on the string distance between their type names, their number of common fields or the values of their fields.



f1(t)	f2(t)	SDI(t)
12	6	0.2
5	6	0.5
4	7	0.5

R	d=0	d=1
f1	-0.99	-
f2	0.5	-

Fig. 1. Features and SDI Analysis

3) Temporal Correlation Between Feature and SDI: In this paper, we only construct features in a supervised way using aggregation over observations. After having grouped observations (for example by observers and/or type of event) we construct subgroups by time intervals. We then apply an aggregate function (such as count, or average over a field) to build time series of features. For illustration purpose, a trivial but significant example of feature is the number of events observed by an agent during a unit of time like illustrated in figure 1. Following this process, we can construct a set of time series of features $\{f_1(t), f_2(t), f_i(t), ..., f_p(t)\}$. Once these time series are built, we can study their delayed correlations with a SDI $g(t)$ over a period of time. We can thus determine the features that might have driven the SDI evolution during this period.

- Let $t \in [1, T]$ be a period of time
- Let $\{f_i(t) \mid i \in [1, P]\}$ be the associated time series
- Let $g(t)$ be an SDI
- Let $d \in [1, D]$ be a delay

We defined the matrix R: $P \times D$ of $r_{i,d}$ as the delay correlation matrix where $r_{i,d}$ is the correlation coefficient between $f_i(t)$ and $g(t + d)$. The final goal is to find the coefficient in the matrix that have the highest magnitude in order to highlight plausible causes of the SDI evolution.

IV. APPLICATION TO DYNAMIC WIRELESS NETWORKS

So far, we have presented a framework to assess systems and to analyze the underlaying factors of this assessment. In this section, we will use this framework to assess dynamic wireless networks. This instantiation narrows the general case to systems where agents are not malicious, use the same SDI functions and share a protocol to exchange information on this SDI.

A. Considered Network Scenarii

The considered system is a wireless mobile ad-hoc network. We implemented it under the Ns3 simulator. Each node has its own mobility model and dynamism. Nodes have a unique wireless interface, might run an UDP server, and instantiate severals UDP On/Off Constant Bit Rate traffic sources. Each UDP source has a destination among the set of server nodes. A source has a fixed data rate and packet size. Duration of activity phasis follows a uniform distribution with fixed bound. We used Ns3 YansWifi Model. Controllers are set in ad-hoc mode and use the adaptive auto rate fallback algorithm without any quality of service. Routes are discovered through the use of AODV. The full list of configurable parameters is described in table III, while a scenario illustration is given figure 2.

B. Framework Instantiation

In this network, nodes are the agents. Each node has a satisfaction function based on the delay it experiences during its communications. It interacts with its environment essentially by its moves and its communications. It can observe others communications and record its own events.

Event, Agent and Observations: Since Ns3 is an event based simulator, it offers interesting properties to instantiate our framework. Among them, its tracing system allows the easy implementation of event observations. In our instantiation, time is a float where the origin is the beginning of the simulation, each observer is an agent, whose id is derived from its IP or MAC address. We have defined various types of

Network Parameters	
N	Number of Nodes
randSeed	Pseudo-random generator initializer
For each node	
X	Initial position on X axis
Y	Initial position on Y axis
hasServer	Implement an UDP server
nbSrc	Number of UDP source
dataRate	Source data rate
pktLen	Packet size
onTime	Min-Max On period duration of sources
offTime	Min-Max Off period duration of sources
noiseFig	Noise figure of the Wifi receiver
mobiModel	Mobility Model (Constant, Random Waypoint, Random Walk)
speed	Min-Max Speed
pause	Min-Max duration of a stable position
xRange	Min-Max position on X-axis
yRange	Min-Max position on Y-axis

TABLE III
WE CAN CONFIGURE SEVERAL PARAMETERS IN OUR NS3 ENVIRONMENT. IN THE NETWORK, EACH NODE HAS AN INITIAL POSITION AND CAN MOVE IN A DEFINED AREA WITH A TUNED MOBILITY. WE CAN ALSO INFLUENCE THE TRAFFIC MATRIX BY CONFIGURING UDP SOURCES AND SERVERS

events, but we can sort them into two main classes: (1) Packet events observable from different nodes (2) Others events, internal to an agent and only accessible by this agent. The latter are : nodes moves, routing table attributes modifications, errors and drops. Table IV delivers further details on the different instantiated types of events.

TABLE IV
EVENT TYPE DESCRIPTION

Type	Information
Packet	Packet capture in promiscuous mode with radiotap header
Rtam	A routing table attribute is modified (number of valid entries, longuest path...)
Move	Speed modification along at least one axis
Ipv4Drop	Packet Drop for a routing reason
PhyRxError	Frame has been received unsuccessfully
PhyRxDrop	Frame dropped during reception
MacTxDrop	Packet dropped before being queued for transmission
MacRxDrop	Packet dropped after the Physical layer
MacTxDataFailed	Data packet transmission failed at mac layer
MacTxRtsFailed	RTS transmission failed at mac layer
MacTxFinal DataFailed	The number of consecutive MacTxDataFailed has reach a threshold
MacTxFinal RtsFailed	The number of consecutive MacTxRtsFailed has reach a threshold

Satisfaction and SDI: In our particular case, the SDI is computed from the observations themselves. We have chosen a fixed aggregation time of 1 second to analyze the network events and construct features. This value is small enough to follow the SDI evolution while sufficiently large to smooth small wireless dynamics. The network assessment is given by the following formulas. Each packet that an UDP source has generated is scored. The scoring function is :

$$score(d) = max\Big(0, \frac{threshold - d}{threshold}\Big) \quad d = delay(p)$$

The score linearly decreases when the delay increases between 0 and a given threshold. It equals 1 for a null delay and 0 if the delay is greater than a threshold (or if the packet is lost). We set the threshold to the arbitrary value of 10ms. A delay associated to a Packet is the timestamps difference between its first observation on the wireless medium and its first observation by its destination. This score is a QoS metric that could be link to the user satisfaction like [8] did. The satisfaction of node i for the interval T is given by the average score for packets that have been generated by i during the interval T:

$$Sat_i(T) = |D|^{-1}.\sum_{d \in D} score(d)$$

$$Sat_i(T) = 1 \quad for \quad |D| = 0$$

$$D = \{delay(p) \mid p.ipSrc = ip(i) \ \cap \ p.time \in T\}$$

The SDI we choose is a simple average of satisfaction over nodes. Thus the SDI for a network of N nodes associated to the interval T is given by:

$$SDI(T) = |N|^{-1}.\sum_{i=1}^{N} Sat_i(T)$$

C. Analysis of Dynamic Wireless Networks

For understanding purpose, we illustrate the analysis of the scenario given in figure 2. This analysis is off-line and the observation is omniscient. As we specified above, we have constructed our features based on aggregation function. Mainly, we grouped observations by observer and by second. We have designed more than twenty features by nodes. Table V details the most relevant ones for this scenario.

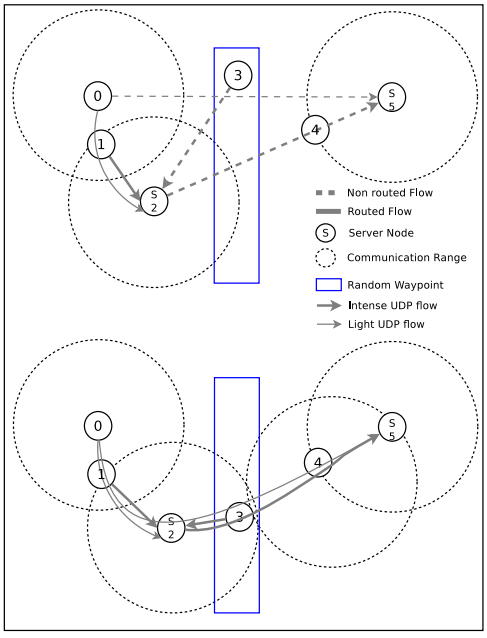


Fig. 2. In this scenario, Nodes 2 and 5 are UDP sinks. Node 3 is mobile. On the top, node 2 might be overloaded. At the bottom, route to 5 is down.

Name	Information
AvgnbGateway	Average # gateway in the routing table
AvgnbValid	Average # valid entry in the routing table
CountPhyRx	# received frame
CountAllRetry	# frame having a retry flag
CountMyRetry	# transmitted frame with a retry flag
CountMyIpFlow	# local distinct IP destination
CountMyUdpSrc	# local active UDP sources
CountAllFlow	# IP flow going through the local node
CountPhyRxError	# PhyRxError events
CountPhyRxDrop	# PhyRxError events
CountDropRouteErr	# IPv4 Drop events for a route error reason

TABLE V
FEATURES DESCRIPTION

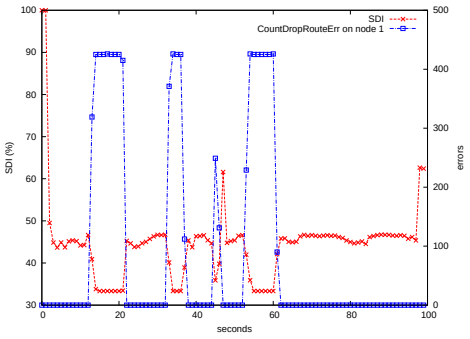
Most relevant constructed features related to the scenario. Each features is related to an observer (called local node). The # stands for *"number of"*

At the beginning of the scenario, all the sources where off, thus all nodes were fully satisfied. Traffic sources started to transmit from second 2 when the SDI brutally decreased. Then, for every significant move of node 3, IP routes are lost or recovered, impacting significantly the SDI. When routes are up, fluctuations can be explained by the delay variations. When nodes 2 experiences some difficulty to transmit, its number of retry will increase and impact the delay. Even if only few packets are concerned, this might have a significant impact on the source satisfaction if these packets are the only ones sent by the source node. Since our SDI takes every nodes in consideration, without any regards on their source volume, we can see an effect on the SDI. After having computed the delayed correlation matrix we found high values for the three features illustrated in figure 3. *CountDropRouteErr* on node 1, *CountPhyRxDrop* on node 5 and *CountMyRetry* on node 2 scores are respectively -0.92, 0.79, -0.88. In figure 3(a), we clearly show that the main fluctuation of the SDI is due to a routing error. Indeed, node 1 can not find a route to node 5 since node 3 has left the path. The retries experienced by node 2 are detailed in figure 3(b). It impacts the SDI when the route is up with a bad communication link between 2 and 3. At first, one can think that transmission retries of node 2 are introduced by the physical drops on node 5. In fact, those events are negatively correlated. Indeed, these nodes can not reach each others due to their relative distances, since the number of drops is much greater than the number of retries, it might come from the fact that node 5 could still be in the carrier range of node 2. Figure 3(c) confirms that node 5 does not drop packets for low SDI, since node 2 does not send them because of routing errors.

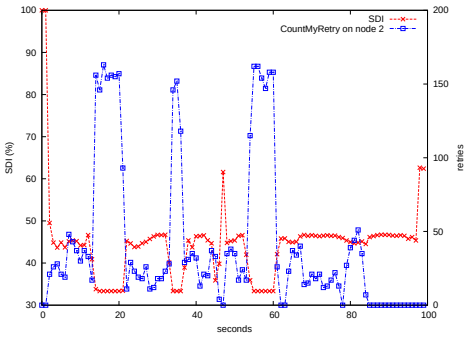
In this case, we led an off-line analysis based on the real value of the SDI. Using simple features based on event counts, we were able to diagnosis the sources of the SDI fluctuation. By construction, these features can be computed in real time by nodes and exchanged to analyze the situation.

V. A COLLABORATIVE SDI ESTIMATION

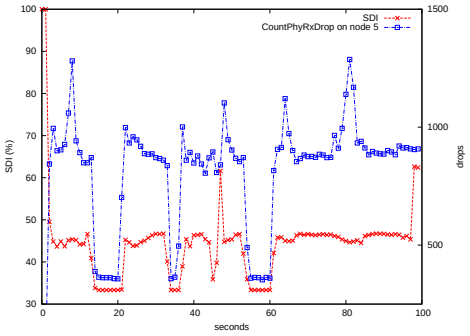
Agents consider their own observations to estimate the SDI value and communicate with their neighbors. We suggest an



(a) CountDropRouteErr on node 1



(b) CountMyRetry on node 2



(c) CountPhyRxDrop on node 5

Fig. 3. Temporal Evolution of SDI Regarding 3 Features

estimation based on existing consensus algorithm. At this point, network layer needs to estimate the satisfaction of upper layers. In our case this comes to evaluate the end-to-end delay experienced by the local application. Thus, in order to estimate the SDI, we first estimate the delay, this estimation will serve to approximate the local satisfaction. Finally we exchange this local satisfaction to have an estimated SDI.

A. Delay Estimation

The end-to-end delay for a packet has already been defined above as the temporal difference between source and destination observations. Each time a packet needs to be forwarded by a node, it waits for a medium access. This time is difficult to predict in our case. Indeed, medium access will be impacted by the level of noise, the number of neighbors, their proximity, their load and the level of interference they produce. As a

result, we will approximate the average delay by a sum of average medium access times. Our approximation will take into account the traffic matrix so that for an interval T, the estimated average delay that a packet can experience when leaving a node i is:

$$\widehat{D}_i(T) = L_i(T) + \sum_{j \in G_i(T)} \rho_j(T).\widehat{D}_j(T-1)$$

$$with \quad \widehat{D}_i(0) = L_i(0)$$

$L_i(T)$ is the average local medium access time, $G_i(T)$ is the set of gateway used by node i during T, $\rho_j(T)$ is the percentage of data traffic sent/forwarded by node i during T that should be forwarded by node j. For each packet, the local processing time equals 0 if the local node is the destination, it equals the threshold value if the packet is dropped, in other cases it is the time between the first observation of the packet from the upcoming link, and the last observation of the packet on the outgoing link. Given a packet arriving at node i, its expected delay is at least the local link process. Depending on its destination, this process time will be added to the expected delay of its destination. The expectation is materialized by ρ, which is a percentage of traffic. The computed average delay for a node will depend on the previous computed value of its neighbors. This implies that nodes have to regularly communicate to update the value of their delay.

B. Local Satisfaction Estimation

Once a node is aware of the average delay it experiences, it is capable to estimate its satisfaction. We approximate the satisfaction by assuming that expectation and scoring function can commute. That is to say (with E_D the expectation function over a set D of delay):

$$E_D(score(d)) \approx score(E_D(d))$$

The approximation error is null when all the observed delays are under the threshold, since the scoring function is linear on the interval [0, threshold]. Thus, we can find a threshold where the approximation can be acceptable. Therefore, we use the average delay $\widehat{D}_i$ to approximate $Sat_i(T)$

$$\widehat{Sat_i}(T) = score(\widehat{D}_i(T)) \approx Sat_i(T)$$

C. SDI Estimation

Since we have defined our SDI as an average, the SDI estimation problem is in fact a dynamic average consensus problem. In our case, the average evolves over time as well as the topology, which is not a fortunate case for previous algorithm. However, despite our problem complexity, satisfaction of nodes over a network are linked in some ways and their dynamics rely on events. Thus the values that compose our average are related and their temporal evolution are driven by the network itself. Therefore we derive an algorithm from existing ones to estimate the SDI value, then we study the impact of satisfaction fluctuation and dynamic topology. In our case, we want to estimate the SDI, which is a time-varying average of the satisfactions. We modified the scheme presented in II-B to suggest the following iteration:

$$\widehat{SDI}_i(t+1) = \alpha_i.\widehat{Sat}_i(t) + \sum_{j \in N_i(t)} \alpha_j.\widehat{SDI}_j(t)$$

$$\widehat{SDI}_i(0) = \widehat{Sat}_i(0)$$

Also if the local satisfaction is known, the real value can be used instead. In this scheme, the satisfaction term allows the consideration of the local satisfaction while the SDI term permits the estimation propagation over the network. The main difference is introduce by $\widehat{Sat}_i(t)$. This term introduce the variability of the satisfaction over time, which was not the case in previous algorithms. The value of α_k could be chosen from the metropolis weight [11] or the maximum degree weight [10].

D. Local Estimation Results

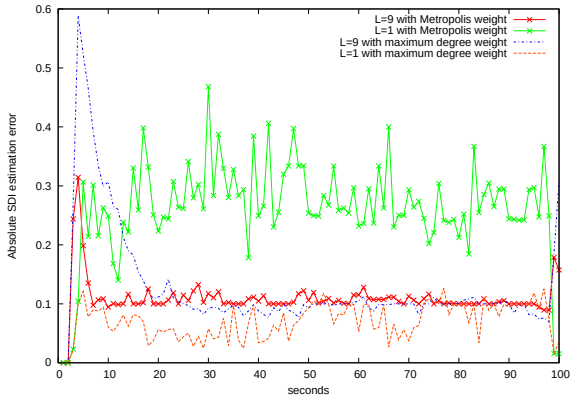
In order to assess the accuracy of our local SDI estimation, we compare estimated SDI values with the real SDI. We conducted 432 Ns3 simulations run to measure the impact of networks characteristics like radius, load and dynamic. We fixed the number of node to ten. Each scenario combines different values of the following parameters : Number of source, Initial average distance, mobility and random seed like illustrated in table VI

Network properties					
Number of Nodes	10				
Source Data Rate	1 Mb/s				
Packet size	1470 Byte				
Source duty cycle	1 (always On)				
Min-Max Speed	5-7				
Random seed	0,1,2				
Number of source (L)	1,4,7,9				
Number of server	10-L				
Initial spacing (d)	20,45,65,75				
Mobility Model	Constant	Random Walk		Random Waypoint	
Area size	-	dxd	d/2xd/2	dxd	d/2xd/2
Pause duration	-	-	-	10, 25, 65	10,25,65

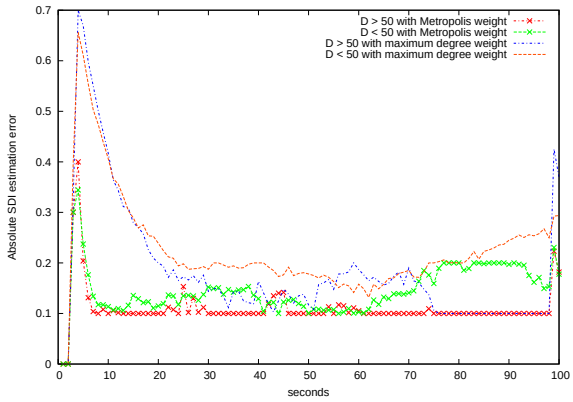
TABLE VI
EACH SIMULATION CONFIGURATION TAKES ITS PARAMETERS BY COMBINING VALUES IN THE TABLE ABOVE

Based on the data set obtained from these simulations, we were able to study the behavior of both weighting algorithm for the average consensus : Metropolis algorithm and Maximum weight algorithm. We compare their properties in different situations. In the first situation we considered a constant topology and two levels of load which were L=1 and L=9. In the second case, we considered an heavy load (L=9) with two levels of mobility. For the first level of mobility, nodes were able to move in a constrained area (D $\leq$ 50), for the second level, nodes were moving in a wider area (D>50). For all scenarii, we computed the absolute value of the difference between the real SDI and the estimated SDI for each node and iteration. To study the convergence rate and the evolution over time, we computed the average of this error. In

figure 4, we plotted the obtained average for all the scenarii in the considered cases.



(a) Impact of load under a constant topology



(b) Impact of mobility under an intense load

Fig. 4. Average error for estimated SDI under several network profiles. For clarity purpose, we did not plot the mid-spread. For information, their values were always under 0.20 for all the curves.

As expected, the initial error tends to be important since nodes have no idea of their neighbor satisfactions. Considering figure 4, because the SDI does not have a constant value, the convergence is not reached. However, the algorithm tends to reduce the estimation error over time. This error converges toward 0.1. In 4(a) like in the case of a constant value, metropolis weight seems more reactive than the maximum degree weight, which, in the case of low traffic might be seen as an over-reaction. When regarding figure 4(b), under an heavy traffic, algorithm appears to better perform under a very dynamic topology. The reasons come from the SDI itself. When the network is too dynamic, routing protocols do not perform well. As a result, the local satisfaction are sufficient not to make mistake on the SDI estimation.

VI. Conclusion and futures work

The invasion of mobile communication in our network have definitely changed their level of complexities. The terminal heterogeneity have multiplied traffic profiles, users are dynamic as well as the topologies. Thus, the management task needs to be mainly delegated to network. In doing so, networks need to evaluate themselves and understand the way they behave. In this paper, our contribution was two-folds. First we introduced a reference model for an observer to assess multi-agent systems and conduct an analysis focused on this assessment. We consider that evaluation are temporal scores, besides we collect event observations to construct time series of features. Our analysis is based on feature correlations to detect which features might have impacted our evaluation. Our model could be applied to distributed systems of several kind as far as they respect few properties. Future work could be lead on the unsupervised way to construct features in order to automate the analysis process. Second, we specifically applied this framework to the self-assessment and self analysis of dynamic wireless networks in the environment of Ns3. After having defined an evaluation policy for our network, we provided a distributed algorithm derived from existing average consensus schemes to compute this assessment. We evaluated this algorithm under various networking conditions to describe its sensitivity to load and topology dynamicity. To improve the algorithm accuracy, future work will be pursued on the use of the designed feature to have a better estimation propagation.

Acknowledgment

This work is partially funded by the French National Research Agency (ANR) under the project ANR VERSO RESCUE (ANR-10-VERS-003)

References

[1] “Cisco visual networking index: Global mobile data traffic forecast update, 2012-2017,” White Paper, Cisco, February 2013.

[2] D. Venmani, Y. Gourhant, L. Reynaud, P. Chemouil, and D. Zeghlache, “Substitution networks based on software defined networking,” in *Ad Hoc Networks*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, J. Zheng, N. Mitton, J. Li, and P. Lorenz, Eds. Springer Berlin Heidelberg, 2013, vol. 111, pp. 242–259. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36958-2_17

[3] The AT&T Labs Research website. [Online]. Available: http://www.research.att.com/projects

[4] E. Bloedorn, A. D. Christiansen, W. Hill, C. Skorupka, L. M. Talbot, and J. Tivel, “Data mining for network intrusion detection: How to get started,” Tech. Rep., 2001.

[5] P. Casas, J. Mazel, and P. Owezarski, “Unsupervised network intrusion detection systems: Detecting the unknown without knowledge,” *Comput. Commun.*, vol. 35, no. 7, pp. 772–783, Apr. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2012.01.016

[6] M. Arlitt and C. Williamson, “An analysis of tcp reset behaviour on the internet,” *ACM SIGCOMM Computer Communication Review*, vol. 35, pp. 37–44, 2005.

[7] B. Rengarajan, “Data mining and coordination to avoid interference in wireless networks.”

[8] S. Ickin, K. Wac, M. Fiedler, L. Janowski, J.-H. Hong, and A. Dey, “Factors influencing quality of experience of commonly used mobile applications,” *Communications Magazine, IEEE*, vol. 50, no. 4, pp. 48–56, 2012.

[9] K. Topley and V. Krishnamurthy, “Average-consensus in a deterministic framework ;part i: Strong connectivity,” *Signal Processing, IEEE Transactions on*, vol. 60, no. 12, pp. 6590–6603, 2012.

[10] L. Xiao, “A scheme for robust distributed sensor fusion based on average consensus,” in *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN*, 2005, pp. 63–70.

[11] L. Xiao, S. Boyd, and S. jean Kim, “Distributed average consensus with least-mean-square deviation,” *Journal of Parallel and Distributed Computing*, vol. 67, pp. 33–46, 2005.