Energy-aware Service Assembly

M. Caporuscio[1], M. D'Angelo[1], V. Grassi[2], and R. Mirandola[3]

[1] Linnaeus University (Sweden), {mauro.caporuscio, mirko.dangelo}@lnu.se
[2] Università di Roma Tor Vergata (Italy), vgrassi@info.uniroma2.it
[3] Politecnico di Milano (Italy), raffaela.mirandola@polimi.it

Abstract.

1 Introduction and Motivation

Contemporary cyber physical systems (for domains like smart cities, intelligent transportation systems, augmented reality) more and more envision the definition of applications that dynamically emerge as opportunistic aggregation of autonomous and independent resources available within the application environment. The service-oriented architecture (SoA) paradigm, in particular its micro-service evolution, appears well suited as reference architectural model for this kind of applications, as it supports the (recursive) vision of new services built as an assembly of independent services, where each service offers specific functionalities, and could require functionalities offered by others to carry out its own task.

However, to be successfully adopted in these emerging computing environment, the building/definition of the services assembly should be able to tackle the following main issues: (i) decentralization: services are offered by autonomous and independent resources distributed in the environment, which makes hardly usable assembly procedures based on the presence of some centralized assembly manager; (ii) dynamics: the offered services are not statically defined but they appear and disappear or change their behavior; (iii) quality-awareness: the assembly should be able to guarantee quality of service (QoS) requirements (e.g., timeliness, availability, cost).

Besides them, another major issue to be considered is: (iv) energy-awareness (efficiency?): the assembly should be able to take into account the energy consumption due to computation and to communication activities. This latter issue is particularly important for several reasons: besides sustainability concerns that are more and more important in the contemporary world, cyber physical systems often rely on battery-powered resources, where a parsimonious and effective use of available energy is mandatory to extend the system lifetime.

Although some effort has been devoted in the past to services assemblies and on their quality of services, only recently ,with the growing interest of the community on sustainability themes, the energy consumption/efficiency? of composite services has received attention/altro Examples can be found in **??**, where service composition of cloud services and of cloud/fog/edge services has been

studied with the goal to minimize the energy consumption by guaranteeing a given QoS. However, to the best of our knowledge, no available solution exists yet able to deal with all the aspects (i)-(iV) mentioned above.

In this paper, we start to build a solution of the following problem:

> how to device a decentralized procedure to dynamically build and maintain over time a fully resolved?functional? assembly of distributed services that are able to collectively fulllfil functional and QoS requirements while minimizing the energy consumption in case of openness and variability of the execution environment.

Specifically, we propose a (fully) decentralized and dynamic service assembly framework whose main characteristics are: (i) a system architecture with explicit modelling of service consumption both for processing and communication activities; (ii) an energy-aware service selection method; (iii) a set of social welfare indexes taking into account quality and energy aspects.

Motivating scenario As a motivating scenario, let us consider the social sensing application (e.g., inspired by [**?**]) shown in Figure 1. Its goal is to infer the social situation of a person (attending a meeting, biking, running, etc.) from data coming from different sensors, and to make it available to people in a given area through social networks so to avoid having too many people in a given space.

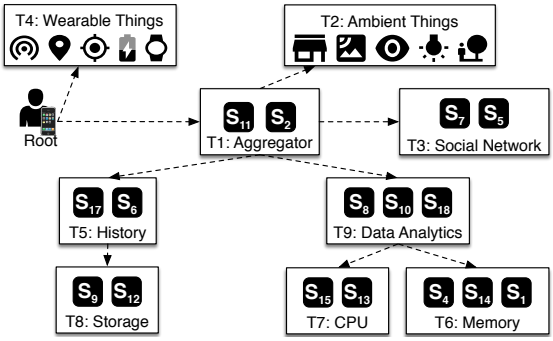


Fig. 1: Motivating Scenario

Raffaela: [*forse possiamo semplificare la figura di FGCS? Io non ho il sorgente*] This application can be built from the aggregation of some logically distinct building blocks that likely include: a *root* module implementing the application's orchestration logic; a set of different sensors providing data about the person's context; one or more compute intensive modules that aggregate data from both these sensors to extract relevant features and draw conclusions about the current situation;

a *Social Network* module to share the current social situation. Besides, these modules require basic *Computing*, and *Communication* resources to fulfill their processing and communication needs, respectively.

We can easily figure out a deployment scenario for this application where all the dependencies expressed by its modules are resolved by services hosted at a single node (typically, some kind of personal device like a smartphone). However, in the envisioned system with autonomous and pervasive resources, several alternatives are likely available. For example, sensor data dependencies can be, at least partially, fulfilled by sensors hosted by other personal devices, or dispersed in the environment. Each such service will be likely interested in experiencing a "good" quality from the services that will be selected to this end with a minimum consumption of energy.

Let us consider a simple example, where the application goal can be achieved by using service S1 and S2 or using S3, like in figure 2.

Each abstract service S_i can be replaced with one of its concrete atomic services (S_{ij}), which are same on function but different with on QoS and Energy consumption. The assembly of concrete services will have to take into account these aspects. Let us suppose for the sake of simplicity, that the QoS requirements are limited to the service execution time. If the attributes of S_{ij} are the one in Table 1 expressed in *time units* and *energy units*, then we can observe that a selection of the service based only on the QoS requirement would lead to the selection of S_{31}, while, taking into account also the energy aspect, we could prefer the combination of S_{11} and S_{21}, which as the minimum energy consumption even if a slight higher execution time.

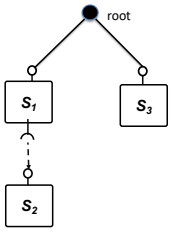


Fig. 2: Service selection example

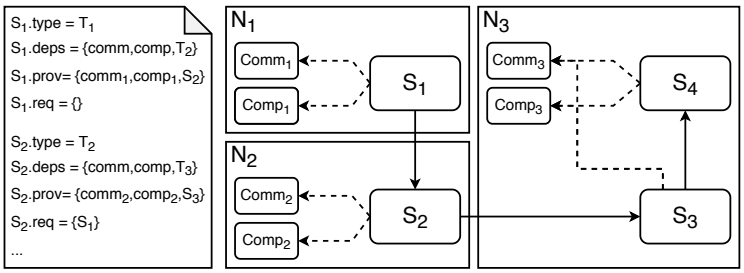
Abstract Service	**Concrete Sevice**	**Quality**	**Comp. Energy**	**Comm. Energy**
S_1	S_{11}	120	5	10
	S_{12}	130	7	15
S_2	S_{21}	50	2	2
S_3	S_{31}	140	10	10
	S_{32}	190	15	10

Table 1: Services attributes

The paper is organized as follows....

2 System model

We consider a set **S** of services, hosted by a set **N** of nodes of a distributed system (e.g. nodes of an edge cloud architecture). We assume that each node consumes energy (to maintain its state when activated and to support the activities of software-implemented services it is hosting), and can possibly produce



green energy (e.g. by wind or solar sources) that can be used to totally or partially compensate its local energy consumption. Each node offers to services it is hosting a set of basic services needed for their execution (e.g., computing, communication and storage services). For the sake of simplicity, in this paper we limit ourselves to consider only computing and communication services offered by a node. Moreover, we also assume that each node offers a single type of computing and communication service. We leave to future work the extension of our model to other basic services categories (e.g.: storage) and to different types of services within each category (e.g., both specialized GPU and general purpose CPU as computing services).

We denote by $compserv(n) \in \mathbf{S}$ and $commserv(n) \in \mathbf{S}$ the computing and communication service offered by a node $n \in \mathbf{N}$, respectively, and by $\mathbf{B} \subseteq \mathbf{S}$ the set of all basic computing and communication services offered by nodes in $\mathbf{N}$ (i.e., $\mathbf{B} = \{S \in \mathbf{S} | S = compserv(n) \vee S = commserv(n)$ for some $n \in \mathbf{N}\}$). Finally, we denote by $node(S) \in \mathbf{N}$ the node hosting service $S \in \mathbf{S}$. [4]

Given these initial definitions, we model a service $S \in \mathbf{S}$ as a tuple $\langle Type, Deps, Prov, Req \rangle$, where:

- $S.Type \in \mathbf{T} = \mathbf{T}' \bigcup \{none, comp, comm\}$ denotes the type of the provided interface (we say that $S.Type$ is the type of S). We assume the existence of a function $match : \mathbf{T} \times \mathbf{T} \rightarrow [0, 1]$ such that $match(T_1, T_2) = 0$ if type T_1 does not match type T_2 and $match(T_1, T_2) > 0$ if a matching exists according to some suitable matching criterion [13,3]. As shown by its definition, the set $\mathbf{T}$ always includes the $comp$ and $comm$ types, with $compserv(n).Type = comp$ and $commserv(n).Type = comm$. Besides them, $\mathbf{T}$ may in general include other types, generically indicated by the set $\mathbf{T}'$.
- $S.Deps \subseteq \mathbf{T}$ is the set of required dependencies for S. We assume that $S.Deps$ is fixed for each service and known in advance. Note that the dependency set $S.Deps$ does not contain duplicates, meaning that a service may depend at most once on any specific interface type in a given context. If $S.Deps = \emptyset$, then S is a *basic* service (i.e. $S \in \mathbf{B}$) that do not need to be connected to other services to carry out its function. On the other hand, for each $S \in \mathbf{S} \setminus \mathbf{B}$ we assume that $S.Deps$ includes at least a dependency $d_1 = comp$

[4] With these definitions, it always holds $node(compserv(n)) = n$ and $node(commserv(n)) = n$.

and a dependency $d_2 = comm$: in this way we model the fact that each non basic service needs at least to be bound to a computing and a communication service, for its internal operations and for its interactions with other services. For each $d \in S.Deps$ we assume that it is known (e.g. through a locally performed monitoring activity) a value $\mu_{S,d}$, which represents the average number of times service S requires dependency d to fulfill each request it has received.
- $S.Prov \subseteq \mathbf{S}$ is the set of *Providers* of S, i.e. the set of services to which S is bound to resolve its dependencies. For services $S \in \mathbf{S} \setminus \mathbf{B}$, we denote by $comp(S) \in \mathbf{B}$ and $comm(S) \in \mathbf{B}$ the computing and communication services used to resolve dependencies $comp \in S.Deps$ and $comm \in S.Deps$, respectively. It must obviously hold $node(comp(S)) = node(comm(S))$ for any $S \in \mathbf{S} \setminus \mathbf{B}$.
- $S.Req \subseteq \mathbf{S}$ is the set of *Requesters* of S, i.e. is the set of other services that are bound to S to resolve one of their dependencies.

A service is either *fully resolved* or *partially resolved*. A service S is *fully resolved* if either: (i) S has no dependencies (i.e., $S \in \mathbf{B}$); or (ii) for all $d \in S.Deps$ there exists a fully resolved service $S' \in S.Prov$ such that $match(d, S'.Type) > 0$.

On the other hand, a *partially resolved* service S has a non-empty list of dependencies, and at least one dependency is either not matched, or is matched by a partially resolved service.

Vincenzo: [*spero di non aver sbagliato con il senso delle frecce e Prov e Req :-) ; controllare*]

A *service assembly* $\mathbf{A}$ is a directed graph $\mathbf{A} = (\mathbf{S}, \mathbf{E})$, where $\mathbf{E} \subseteq \mathbf{S} \times \mathbf{S}$ is the set of resolved dependencies. Specifically, a directed edge $(S_i, S_j) \in \mathbf{E}$ denotes that S_i is using S_j to resolve one of its dependencies. In general, a given S_i has multiple simultaneous outgoing bindings (towards services in $S_i.Prov$), one for each dependency, and can have multiple simultaneous incoming bindings from other services (belonging to $S_i.Req$), using S_i to resolve one of their dependencies.

According to this model, the problem introduced above can be restated as:
Vincenzo: [*da rivedere*]

> *how to dynamically build and maintain over time a fully resolved assembly of distributed services that are able to collectively fulfill functional, QoS and energy requirements in case of openness, variability, unpredictability and scalability of the execution environment.*

3 Energy model

In this section we introduce the model we adopt to estimate the energy consumption of each service, as a function of the bindings it establishes with other services to resolve its dependencies. As we are considering only computing and

communication resources as the "physical"resources supporting the service operations, the model consists of two parts: a *computation?computing? energy* model and a *communication energy* model that describe, respectively, the energy consumption caused by the computing and communication activities triggered by a service S to fulfill each request it receives.

3.1 Computation energy

Let us consider a service $S \in \mathbf{S} \setminus \mathbf{B}$. We recall that S represents in this case a "virtual" (software implemented) resource that generates (depending on its input flow) flows of requests to all services in $S.Prov$ used to resolve its dependencies. When the flows of requests addressed to services in $S.Prov$ eventually reach a service of type *comp*, then they will cause some computation energy consumption. This will happen in one step for the dependency of type *comp* of S ("internal operations" of S). Otherwise, the flow of requests will go through a number of virtual services before reaching a service of type *comp*. To model this process, we introduce the following three indexes $S.I^{comp}$, $S.L^{comp}$ and $S.E^{comp}$ that model, respectively, the *individual*, *node level* and *system level* computation energy consumption associated with?caused by? a single request addressed to S. They are defined as follows:

$$S.I^{comp} = h_{node(S)}(\mu_{S,comp}) \tag{1}$$

$$S.L^{comp} = h_{node(S)}(\mu_{S,comp}) + \sum_{\substack{S' \in S.Prov, \\ s.t. S'.Type \neq comp \\ \wedge node(S') = node(S)}} \mu_{S,S'.Type} \cdot S'.L^{comp} \tag{2}$$

$$S.E^{comp} = h_{node(S)}(\mu_{S,comp}) + \sum_{\substack{S' \in S.Prov, \\ s.t. S'.Type \neq comp}} \mu_{S,S'.Type} \cdot S'.E^{comp} \tag{3}$$

where $h_n(\mu)$ represents the energy consumption of the *comp* service hosted by a node n for the execution of μ operations. **Vincenzo:** [*dire che stiamo convenzionalmente assumendo una "operazione media" come unita' di computazione?*] . As an example, $h_n(\mu)$ could be instantiated as follows:

$$h_n(\mu) = a_S + e_S \cdot \mu \tag{4}$$

where we adopt a linear computation energy that consists of a *fixed* part a_S, and a *dynamic* part $e_S \cdot \mu$ linearly depending on the load addressed to the service of type *comp*. **Vincenzo:** [*aggiungere dettagli? tipo che a_S modella il consumo causato dal semplice fatto che S e' switched on, mentre e_S e' il consumo di una "unita' di computazione"; e μ quindi rappresenta il numero di "unita' di computazione" da eseguire*]

From the definitions given above, we see that $S.I^{comp}$ models only the energy consumption directly consumed by S for its internal operations. Besides the directly consumed energy, $S.L^{comp}$ includes also the computation energy indirectly consumed by S because of its use of services $S' \in S.Prov$, but limited to services co-located with S on the same node (their energy consumption is multiplied by the average number of times S uses S', given by $\mu_{S,S'.Type}$). Finally, $S.E^{comp}$ adopts a system-wide perspective and models the overall computation energy consumption caused by S on any node in the system.

We point out that $S.I^{comp}$, $S.L^{comp}$ and $S.E^{comp}$ refer to computing?computation? energy consumption originated from?caused by? a single request addressed to S. To get measures of the energy consumption per unit time (energy consumption rate), we introduce the concept of *load vector* $\mathbf{\Lambda}_S = [\lambda_S(n)]_{n \in \mathbf{N}}$ associated with any service $S \in \mathbf{S}$, where each vector entry $\lambda_S(n)$ denotes a flow of requests (expressed as requests per unit time) addressed to service S by services hosted by node $n \in \mathbf{N}$. $\mathbf{\Lambda}_S$ can be easily estimated by some local monitoring activity. Given a load vector Λ_S, we can derive from $S.I^{comp}$, $S.L^{comp}$ and $S.E^{comp}$ corresponding measures of the computing energy consumption rate **Vincenzo:** [*giusto?*]

:

$$S.\rho_X^{comp} = \Big(\sum_{n \in \mathbf{N}} \lambda_S(n) \Big) \cdot S.X^{comp} \tag{5}$$

where X stands for any of: I, L, E.

3.2 Communication Energy

Let us consider a service $S \in \mathbf{S} \setminus \mathbf{B}$. As already stated in the previous subsection, S represents a "virtual" (software implemented) resource: the communication energy consumption caused by S depends on the interactions that S has both with services that use it to resolve their dependencies (services in the set $S.Req$) and services used by S itself to resolve its dependencies (services in the set $S.Prov$). Actually, it corresponds to the energy consumed by the communication services hosted by the system nodes to support?implement? the interactions directly of indirectly triggered by S, when they involve services hosted by different nodes. In this respect, we assume that the energy spent by a communication service of a node n for these interactions depends both on their volume, and the latency and bandwidth of the links connecting it with other nodes. **Vincenzo:** [*mettere rif?*]

To model this process, we introduce the following three indexes $S.I_n^{comm}$, $S.L_n^{comm}$ and $S.E_n^{comm}$ that model, respectively, the *individual*, *node level* and *system level* communication energy consumption associated with?caused by? a single request addressed to S by some other service hosted by a node $n \in \mathbf{N}$. They are defined as follows:

$$S.I_n^{comm} = \phi^{req}_{node(S)}(\delta_S^{rcv}, bw(n, node(S)), lt(n, node(S)))$$
$$+ \sum_{\substack{S' \in S.Prov, \\ s.t. node(S) \neq node(S')}} \mu_{S,S'.Type} \cdot \phi^{prov}_{node(S)}(\delta^{snd}_{S,S'.Type}, bw(node(S), node(S')), lt(node(S), node(S'))) \tag{6}$$

$$S.L_n^{comm} = S.I_n^{comm} + \sum_{\substack{S' \in S.Prov, \\ s.t. node(S') = node(S)}} \mu_{S,S'.Type} \cdot S'.L^{comm}_{node(S)} \tag{7}$$

$$S.E_n^{comm} = S.I_n^{comm} + \sum_{S' \in S.Prov} \mu_{S,S'.Type} \cdot S'.E^{comm}_{node(S)} \tag{8}$$

where:

- $bw(n_1, n_2)$ and $lt(n_1, n_2)$, with $n_1, n_2 \in \mathbf{N}$, denote, respectively, the bandwidth and latency of the link connecting nodes n_1 and n_2;
- δ_S^{rcv} and $\delta^{snd}_{S,d}$ denote, respectively, the average amount of data S receives for each service request addressed to it, and the average amount of data S sends for each invocation of its dependency d, to fulfill that request;
- $\phi_n^{req}(\delta, b, l)$ denotes the energy consumed by $commserv(n)$, $n \in \mathbf{N}$, when it receives an amount δ of data **Vincenzo:** [*expressed as "communication units"?*] addressed to a service hosted by n over a link with bandwidth b and latency l;
- $\phi_n^{prov}(\delta, b, l)$ denote the energy consumed by $commserv(n)$, $n \in \mathbf{N}$, when it sends an amount δ of data **Vincenzo:** [*expressed as "communication units" ?*] to a service hosted by another node over a link with bandwidth b and latency l; **Vincenzo:** [*dire che diamo piu' avanti (dove?) dettagli su possibili modi di definire le ϕ?*]

The first term in the r.h.s. of equation (6) represents the energy consumed by $commserv(node(S))$ for the reception of the service request addressed to S, coming from an external node n. The second term in the r.h.s. of equation (6) represents the energy consumed by $commserv(node(S))$ to send the requests S addresses to services solving its dependencies (i.e., services in $S.Prov$) and hosted by different nodes. Hence, $S.I_n^{comm}$ models the communication energy consumption of $commserv(node(S))$ caused only by the direct interactions S has with other services hosted by different nodes.

On the other hand, $S.L_n^{comm}$ in equation (7) adds to the energy consumption measured by $S.I_n^{comm}$ also the communication energy consumption indirectly caused by S, corresponding to interactions that services in $S.Prov$ have

with other services to carry out their own task. As it can be seen from the r.h.s. of equation (7), $S.L_n^{comm}$ limits its scope to the energy consumption of $commserv(node(S))$ only.

Finally, $S.E_n^{comm}$ in equation (17) adopts a system-wide perspective, measuring the communication energy consumption directly or indirectly caused by S on any node in the system, when S receives a single from a service hosted by a node n.

Vincenzo: [*questo modello consente di distinguere tra flussi in ingresso (che vengono da $S.Req$) e in uscita (diretti verso $S.Prov$), con conseguenti possibili diversi costi energetici (definendo opportunamente le due ϕ; pero' non tiene conto del fatto che (in scenari client-server) messaggi in ingresso comportano anche messaggi in uscita, e viceversa; per tenerne conto, bisognerebbe complicare un po' la notazione, da vedere se ne vale la pena; oppure immaginare che questi aspetti siano incorporati nelle ϕ?*]

Analogously to the computing energy case, we can derive from $S.E_n^{comm}$, $S.I_n^{comm}$ and $S.L_n^{comm}$ measures of the communication energy consumption rate, given a load vector Λ_S:

$$S.\rho_X^{comm} = \sum_{n \in \mathbf{N}} \lambda_S(n) \cdot S.X_n^{comm} \tag{9}$$

where X stands for any of: E, I, L.

4 Social Welfare Indexes

Vincenzo: [*se eliminiamo la sez di QoS, c'e' da capire come si introducono gli indici globali di QoS*]

In this section we formally define the indexes, based on the model defined in the previous sections, that we will use to measure the effectiveness of our approach with respect to its ability in achieving a good local and social welfare. In particular, with regard to the latter, we adopt a two-fold perspective. On the one side, we measure the average global system "quality" (with respect to both QoS and energy consumption) achieved thanks to the contribution of all services. On the other side, we measure whether there is an unbalanced distribution among services of this global quality. The adopted measures are defined as follows.

Given a fully resolved assembly $\mathbf{A} = \mathbf{S} \times \mathbf{E}$, let us consider first its global system quality. From the QoS perspective, we define as follows the *Global QoS* delivered by all services in $\mathbf{A}$:

$$GQoS(\mathbf{A}) = \frac{1}{|\mathbf{S}|} \sum_{S \in \mathbf{S}} S.Q \tag{10}$$

where $S.Q$ is defined as in Equation (**??**).

From the energy consumption perspective, we define as follows the *Global Energy Balance* delivered by all services in $\mathbf{A}$:

$$GEB(\mathbf{A}) = \frac{1}{|\mathbf{N}|} \sum_{n \in \mathbf{N}} \Big(G_n - \sum_{\substack{S \in \mathbf{S} \\ s.t. node(S)=n}} (S.\rho_I^{comp} + S.\rho_I^{comm}) \Big) \tag{11}$$

Vincenzo: [*nota: eq. 11 assume implicitamente che l'energia green in eccesso prodotta da un nodo possa compensare i consumi di altri nodi; se questo non fosse vero (se cioe' l'energia green prodotta localmente puo' solo essere consumata in loco), eq. 11 andrebbe riscritta come segue:*]

$$GEB(\mathbf{A}) = \frac{1}{|\mathbf{N}|} \sum_{n \in \mathbf{N}} \min \Big\{ 0, \Big(G_n - \sum_{\substack{S \in \mathbf{S} \\ s.t. node(S)=n}} (S.\rho_I^{comp} + S.\rho_I^{comm}) \Big) \Big\} \tag{12}$$

where G_n in the green energy generation rate of node $n \in \mathbf{N}$, and $S.\rho_I^{comp}$ and $S.\rho_I^{comm}$ are defined as in Equations (5) and (9), respectively. We use $S.\rho_I^{comp}$ and $S.\rho_I^{comm}$ in the definition of $GEB(\mathbf{A})$ to avoid counting more than once the energy consumption caused by a service. **Vincenzo:** [*e' giusto?*]

We remark that both $GQoS(\mathbf{A})$ and $GEB(\mathbf{A})$ are "higher is better" indexes. However, they do not allow to capture to what extent all involved services fairly share?contribute to? the measured average quality. To this end, we introduce the following *fairness* indexes based on the *Jain's fairness index* [**?**] as additional measures of the achieved social welfare, to measure how uniform is the quality achieved by all the participating services, from the QoS and energy consumption perspectives, respectively:

Vincenzo: [*per indice fairness: considerare come alternativa a Jain il criterio di massimizzazione della qualita' (QoS, energia) peggiore ?*]

$$FQoS(\mathbf{A}) = \frac{(\sum_{S \in \mathbf{S}} S.Q)^2}{|\mathbf{S}| \sum_{S \in \mathbf{S}} S.Q^2} \tag{13}$$

$$FEB(\mathbf{A}) = \frac{\Big(\sum_{n \in \mathbf{N}} \Big(G_n - \sum_{\substack{S \in \mathbf{S} \\ s.t. node(S)=n}} (S.\rho_I^{comp} + S.\rho_I^{comm}) \Big) \Big)^2}{|\mathbf{N}| \sum_{n \in \mathbf{N}} \Big(G_n - \sum_{\substack{S \in \mathbf{S} \\ s.t. node(S)=n}} (S.\rho_I^{comp} + S.\rho_I^{comm}) \Big)^2} \tag{14}$$

The value of these fairness indexes ranges from $\frac{1}{|\mathbf{S}|}$ or $\frac{1}{|\mathbf{N}|}$, respectively (worst case), to 1 (best case), and it is maximum when all services deliver the same quality (QoS or energy consumption) level. In general, indexes of this type penalize situations where the quality achieved by different services is highly unbalanced. Hence, by using $FQoS(\mathbf{A})$ or $FEB(\mathbf{A})$, we intend to reward assemblies that result in a fair share of the overall quality measured by $GQoS(\mathbf{A})$ and $GEB(\mathbf{A})$, respectively.

5 Energy(Power?)-aware service selection

Vincenzo: [*Commenti generali sui criteri di selezione QoS-aware e energy-aware riportati nel seguito. Nel caso della selezione QoS-aware, per come e' scritta ora, e' una "banale" tecnica di selezione greedy; se la QoS non e' load-dependent immagino sia la selezione ottima; se la QoS e' load-dependent, bisogna evidentemente evitare la corsa di tutti verso il servizio che appare attualmente "migliore", e quindi il criterio scritto sopra andrebbe precisato; qui entrano in gioco due fattori (vedi lavoro di Shaerf): 1) come si costruisce una stima di* $S_i.Q$*, sulla base di osservazioni fatte e/o informazioni ricevute (tra i fattori che entrano in gioco qui: quale peso dare a informazioni piu' o meno recenti; quale peso dare a informazioni acquisite direttamente o ricevute da terzi); 2) una volta ottenute le stime di* $S_i.Q$*, come selezionare* $\overline{S}$*: qui entra in gioco la questione exploitation vs. exploration. Nel caso della selezione energy-aware, mi sembra che in effetti, nel nostro modello, non c'e' load-dependence: nel nostro modello il mio consumo cresce con il carico che io sottopongo, ma il consumo che vedo io non e' influenzato dai consumi degli altri. Con questo modello, quindi, un criterio greedy direi che e' ottimale, e in effetti i due criteri riportati sotto li qualificherei come greedy: differiscono solo per il tipo di informazione utilizzata (locale, che non necessita di advertisament, e globale, che invece necessita di advertisement (con conseguenti problematiche di trust?)*]

We are assuming that each service S advertises its QoS by publishing the current value of the QoS index $S.Q$, known through an internal monitoring activity, as described *where*???. Moreover, S could also advertise (see later ....) its system-wide energy consumption by publishing the current values of $S.E^{comp}$ and $S.E_n^{comm}$, $n \in \mathbf{N}$, estimated according to equations (3) and (8), respectively.

Let us assume that S must select a service within a set of candidates $\Omega_{S,d} = \{S_1, S_2, \ldots, S_k\}$ to resolve its dependency $d \in S.Deps$.

QoS-driven selection A possible QoS-driven criterion to drive the selection could be defined as follows:

– select $\overline{S} \in \Omega_{S,d}$ such that:

$$\overline{S} = \arg\max_{S_i \in \Omega_{S,d}} \{S_i.Q\} \tag{15}$$

Energy-driven selection Possible energy-driven criteria to drive the selection could be defined as follows:

1. select $\overline{S} \in \Omega_{S,d}$ such that:

$$\overline{S} = \arg\min_{S_i \in \Omega_{S,d}} \{S_i.E^{comp} + \mu_{S,S_i.Type} \cdot S_i.E^{comm}_{node(S)}\} \tag{16}$$

The use of this criterion implies that each service S advertises the values $S.E^{comp}$ and $S.E_n^{comm}$, $n \in \mathbf{N}$;

2. select $\overline{S} \in \Omega_{S,d}$ such that:

$$\overline{S} = \arg\min_{S' \in \Omega_{S,d}} \Big\{ \mu_{S,S'.Type} \cdot S'.L^{comp} \cdot I_{\{node(S')=node(S)\}} + \mu_{S,S'.Type} \cdot \phi^{prov}_{node(S)}(\delta^{snd}_{S,S'.Type}, bw(node(S), node(S')), lt(node(S), node(S')) \cdot I_{\{node(S') \neq node(S)\}} + \mu_{S,S'.Type} \cdot S'.L^{comm}_{node(S)} \cdot I_{\{node(S')=node(S)\}} \Big\} \quad (17)$$

Vincenzo: [*spiegare i vari termini di questa espressione? fare qualche commento sulla differenza tra questo criterio e il precedente?*]

where $I_{\{cond\}}$ is the indicator function that holds 1 when condition *cond* is true, and 0 otherwise. The use of this criterion does not require the advertisement of any energy consumption value, as all the information needed for its application can be collected at each node by a local monitoring activity; **Vincenzo:** [*gli indici $S.L^{comp}$ e $S.L^{comm}_n$ sono basati solo su informazioni locali, quindi in effetti non richiedono che venga fatta circolare nessun tipo di informazione; potrebbe essere interessante vedere che cosa succede agli indici di sistema quando si usa un criterio di selezione basato solo su informazione locale rispetto a uno dove viene fatta circolare informazione (usando indici come $S.E^{comp}$ e $S.E^{comm}_n$), che e' un po' la discussione che faceva Schaerf nel suo lavoro su RL e load balancing*]

3. select $\overline{S} \in \Omega_{S,d}$ such that:

$$node(\overline{S}) = \arg\max_{n \in node(\Omega_{S,d})} \{EB_n\} \quad (18)$$

where, with a little abuse of notation, $node(\Omega_{S,d}) \subseteq \mathbf{N}$ denotes the set of all nodes hosting services that belongs to the set $\Omega_{S,d}$, while EB_n denotes the *energy balance* of node n, defined as follows:

$$EB_n = G_n - \sum_{\substack{S \in \mathbf{S} \\ s.t. node(S)=n}} (S.\rho^{comp}_I + S.\rho^{comm}_I) \quad (19)$$

Mirko: [*la definizione di energy balance introdotta prima dei social welfare indexes alleggerirebbe la notazione*]

or:

$$EB_n = \min \Big\{ 0, \Big(G_n - \sum_{\substack{S \in \mathbf{S} \\ s.t. node(S)=n}} (S.\rho^{comp}_I + S.\rho^{comm}_I) \Big) \Big\} \quad (20)$$

depending on whether or not we can assume the existence of a power grid (?) among system nodes, as discussed in Section ... . We assume that the EB_n value is locally estimated at each node, and advertised by all services belonging to that node. The use of this criterion aims at fairly distributing

energy consumption among the system nodes, by selecting the service in $\Omega_{S,d}$ hosted by the node that currently results to have the best energy balance. **Vincenzo:** [*questo criterio, a differenza dei due precedenti, introduce problematiche di load balancing: se tutti scelgono servizi appartenenti al nodo con il miglior bilancio energetico, il bilancio di questo nodo peggiorera' sensibilmente, etc. ...; quindi il suo uso richiederebbe meccanismi tipo 2HRL, o simili*]

4. Learning-based fair strategy:
   Each service S_i advertises its overall energy consumption.

$$\overline{S_i.E} = S_i.E^{comp} + \mu_{S,S_i.Type} \cdot S_i.E^{comm}_{node(S)} \tag{21}$$

Adopting a greedy selection strategy on this index (Equation 16) may generate unfair assemblies. Inspired by Shaerf et. al. [15] we adopt a multi-agent reinforcement learning strategy which aims at balancing the energy consumption in the system and generate energy efficient assemblies that are also fair.
To this end, each service S_j maintains an efficiency estimator $ee(S_j)$ for the overall consumption generated by a service S_j. This estimator is a weighted sum of the overall energy consumption declared (at each interaction) by S_j.

$$ee(S_j) = W \times \overline{S_i.E} + (1 - W) \times ee(S_j) \tag{22}$$

The efficiency estimator is updated at each interaction as in Equation 16, where W is a value in the range $[0, 1]$ that depends on the number of times a service S_i interacted with a service S_j, i.e., n_{S_j}. Intuitively, W decreases with increasing n_{S_j}, by giving more importance to the history and not to new values.
The service selection function adopts a probabilistic approach for the selection of a service S_j. First we define Equation 23:

$$\overline{p(S_j)} = \begin{cases} ee(S_j)^{-n} & \text{if } n_{S_j} > 0 \\ A[ee(S)]^{-n} & \text{if } n_{S_j} = 0 \end{cases} \tag{23}$$

Where A[ee(S)] represent the average of the overall values $ee(S_j)$ for each S_j with $n_{S_j} > 0$ and n is a positive real-valued parameter. Finally, the probability of selecting a service S_j is defined as in Equation 4:

$$p(S_j) = \frac{\overline{p(S_j)}}{\sigma} \tag{24}$$

Where σ is a normalization factor, $\sum_{S_j in S} \overline{p(S_j)}$[5]. If a service do not have experiences in the vector $ee(S)$ it selects a service according to the local-rule, Equation 17.

[5] For a comprehensive discussion on how to setup the parameters n, W we refer the interested reader to [15] and to our replication package for the actual instantiation in our scenario.

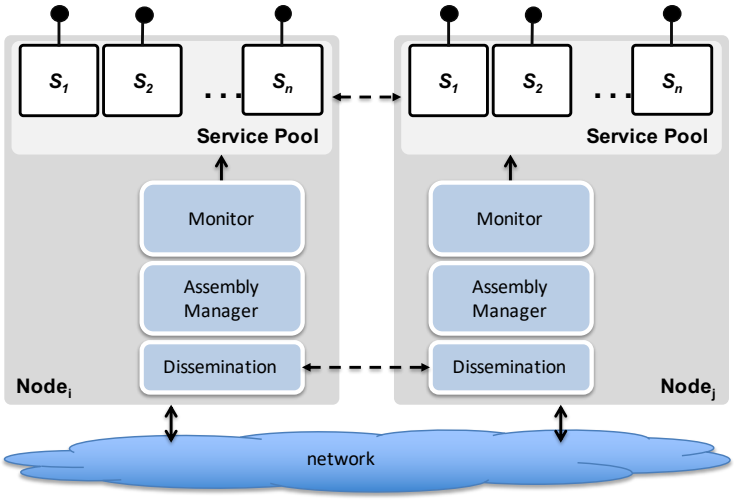


Fig. 3: Node Architecture

6 System Architecture

The goal is to drive the service-oriented system we are considering towards the construction of a service assembly that is energy efficient (see in Section 4). In this section we present a fully decentralized architecture allowing for achieving this goal.

Figure 3 shows the set of macrocomponents deployed at each $Node_i$: *Service Pool* is the set of services $S \in \mathbf{S}$ running on $Node_i$. *Monitor* is in charge of monitoring the energy consumption (i.e., $S.X^{comp}$ and $S.X^{comm}$) for each service S in the Service Pool. *Assembly Manager* receives from *Dissemination* the set $S.Prov$ that specifies which services should currently be used to solve the dependencies of a local service S, and manages the corresponding bindings. Moreover, it receives notifications of incoming binding requests for each local service S, and keeps updated the corresponding set $S.Req$. Finally, *Dissemination* implements decentralized information dissemination by exploiting a gossip communication model [9,16].

7 Experimental Evaluation

In this section we present a set of simulation experiments to assess... ? . The experiments aim at ... ?. To this end, we implemented a large-scale simulation model for the PeerSim simulator [12].

We experiment with a deployment scenario in line with social sensing application presented in [6], where the goal is to infer the social situation of a person from data coming from different sensors. We mimic a wireless sensor network

Algorithm 1 UPDATE() for scalar **U**()

// Input parameters
$Best_{S,d} \subseteq \mathbf{S}; S_i \in \mathbf{S}$;
// Algorithm
$Best_{S,d} \leftarrow Best_{S,d} \cup \{S_i\}$
if $|Best_{S,d}| \leq H$ **then**
    continue
else/* drop the worst service to keep $|Best_{S,d}| \leq H$ */
    $jmin \leftarrow \arg\min_j \{S.X_j^{comp/comm} \mid S_j \in Best_{S,d}\}$
    $Best_{S,d} \leftarrow Best_{S,d} \setminus \{S_{jmin}\}$
return $Best_{S,d}$

Algorithm 2 Algorithm executed by the agent responsible for service S

// Input parameters
S; **U**(); SELECT(); UPDATE(), (for all $d \in S.Deps$)
// Local variables
$S.Prov \leftarrow \emptyset$
$Best_{S,d} \leftarrow \emptyset$; for all $d \in S.Deps$

procedure ACTIVETHREAD
    loop
        Wait Δt
        for all $S_j \in$ GETPEERS() **do**
            Send $\langle S.Prov \cup \{S\} \rangle$ to S_j

procedure PASSIVETHREAD
    loop
        Wait for message $\langle \mathbf{M} \rangle$ from S_j
        for all $S_k \in \mathbf{M}$ **do**
            if there exists $d \in S.Deps$ s.t. $matches(d, S_k.Type) > 0$ **then**
                $Best_{S,d} \leftarrow$ UPDATE$(Best_{S,d}; S_k; \mathbf{U}())$
                $S.Prov \leftarrow$ SELECT$(Best_{S,d}; S.Prov; \mathbf{U}())$

(WSN) deployment scenario of an edge computing application running in an university/campus, stadium or shopping mall.

7.1 Experiment setting

In classsic WSN applications, the energy cost to transmit one bit is typically around 500-1000 times greater than a single 32-bit computation [1,17] (i.e., for the same amount of bits, the communication energy cost is one to two order of magnitude bigger than the computation energy cost). Thus, for applications with simple functionality, striving for CPU energy-efficiency might not be worthwhile [21]. However, with the increasing development of more advanced, computationally intensive applications at the edge of the network, the energy consumption for CPU operations is playing an important role on the total energy

footprint of the system. Hence, with complex applications deployed in WSN, it is reasonable to consider the computation and communication costs equally relevant. Then in the experimentation, we define the cost of a k-bit CPU operation in the network equal to the average cost of a k-bit transmission.

We simulate communication thought state of the art wireless technology (e.g., BT 5.2, 5G/6G) and deploy the services in an network area with diameter of 200 meters. We adopt the latency-centric model described in Section **??** as communication energy consumption model. In the identified scenario, and for the considered network diameter, the energy cost of a k-bit transmission is on average two times more costly than the energy cost of receiving k-bit.

The nodes are randomly positioned in the area with symmetric latency links and are endowed with the same energy generation rate. Without loss of generality, we assume that the packet loss in the network is null and that each node in the network hosts a single service and offers basic computing and communication functionalities.

7.2 Greedy selection

7.3 Fair selection

Mirko: [*come selezionare fair?*]

8 Related Work

In [8] there is a table summarizing the work done in energy saving/efficiency from a software point of view.This can be helpful.

In [22] the energy demand of a server (i.e., fog/edge node) consists of static and dynamic parts. The static part is fixed once the server is activated, regardless of the resource utilization. While the dynamic part is proportional to the resource utilization (i.e., how many services a physical node hosts) [22]. In our previous paper [6], we did not aim at measuring the impact of multiple services on single nodes. Without loss of generality we assumed that each node hosted a single service. Moreover, we did not take into account replication of services. Because of this, if we assume that all the services have the same resource requirement, our computation energy (static plus dynamic) according to this model is the same for every node of the platform.

Mirko: [*Differentiate between works on service placement on fog/edge (cite some) and service assembly on fog/edge (should we look more RW respect our journal?). In this work we assume that the placement of services in the infrastructure is done in advance and we aim at assemble them*]

9 Conclusion

9.1 Future Work

Mirko: [*additinal experimentation: definire classi di applicazioni su cui fare la sperimentazione. Determinate applicazioni sono ad esempio data intensive and latency sensitive (e.g., IoT), altre compute intensive (e.g., computational offload of heavy tasks). Queste classi potrebbero caratterizzare la scelta di alcuni parametri e modelli nelle sperimentazioni. In questo lavoro preliminare abbiamo lavorato solo con una specifica applicazione*]

References

1. Barr, K.C., Asanović, K.: Energy-aware lossless data compression. ACM Trans. Comput. Syst. **24**(3), 250–291 (Aug 2006). https://doi.org/10.1145/1151690.1151692
2. Beaumont, O., Eyraud-Dubois, L., Won, Y.J.: Using the last-mile model as a distributed scheme for available bandwidth prediction. In: Jeannot, E., Namyst, R., Roman, J. (eds.) Euro-Par 2011 Parallel Processing. pp. 103–116. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
3. Caporuscio, M., Ghezzi, C.: Engineering future internet applications: The prime approach. Journal of Systems and Software **106**, 9 – 27 (2015). https://doi.org/https://doi.org/10.1016/j.jss.2015.03.102
4. Cardellini, V., D'Angelo, M., Grassi, V., Marzolla, M., Mirandola, R.: A decentralized approach to network-aware service composition. In: Dustdar, S., Leymann, F., Villari, M. (eds.) Service Oriented and Cloud Computing. pp. 34–48. Springer International Publishing, Cham (2015)
5. Dabek, F., Cox, R., Kaashoek, F., Morris, R.: Vivaldi: A decentralized network coordinate system. In: Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. p. 15–26. SIGCOMM '04, Association for Computing Machinery, New York, NY, USA (2004). https://doi.org/10.1145/1015467.1015471
6. D'Angelo, M., Caporuscio, M., Grassi, V., Mirandola, R.: Decentralized learning for self-adaptive qos-aware service assembly. Future Generation Computer Systems **108**, 210 – 227 (2020). https://doi.org/https://doi.org/10.1016/j.future.2020.02.027
7. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences. pp. 10 pp. vol.2– (2000)
8. Horcas, J.M., Pinto, M., Fuentes, L.: Context-aware energy-efficient applications for cyber-physical systems. Ad Hoc Networks **82**, 15 – 30 (2019). https://doi.org/https://doi.org/10.1016/j.adhoc.2018.08.004, http://www.sciencedirect.com/science/article/pii/S1570870518305572
9. Jelasity, M., Montresor, A., Babaoglu, Ö.: Gossip-based aggregation in large dynamic networks. ACM Trans. Comput. Syst. **23**(3), 219–252 (2005)
10. Kumar, K., Lu, Y.H.: Cloud computing for mobile users: Can offloading computation save energy? Computer **43**(4), 51–56 (2010)

11. Liao, Y., Geurts, P., Leduc, G.: Network distance prediction based on decentralized matrix factorization. In: Crovella, M., Feeney, L.M., Rubenstein, D., Raghavan, S.V. (eds.) NETWORKING 2010. pp. 15–26. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
12. Montresor, A., Jelasity, M.: Peersim: A scalable p2p simulator. In: 2009 IEEE Ninth International Conference on Peer-to-Peer Computing. pp. 99–100 (2009)
13. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of web services capabilities. pp. 333–347 (2002)
14. Ramasubramanian, V., Malkhi, D., Kuhn, F., Abraham, I., Balakrishnan, M., Gupta, A., Akella, A.: A unified network coordinate system for bandwidth and latency. Technical Report MSR-TR-2008-124 (2008)
15. Schaerf, A., Shoham, Y., Tennenholtz, M.: Adaptive load balancing: A study in multi-agent learning. J. Artif. Int. Res. **2**(1), 475–500 (May 1995)
16. Shah, D.: Gossip Algorithms. Foundations and trends in networking, Now Publishers (2009)
17. Singh, M., Prasanna, V.K.: System-Level Energy Tradeoffs for Collaborative Computation in Wireless Networks, pp. 113–131. Springer US, Boston, MA (2002)
18. Song, S., Keleher, P., Bhattacharjee, B., Sussman, A.: Decentralized, accurate, and low-cost network bandwidth prediction. In: 2011 Proceedings IEEE INFOCOM. pp. 6–10 (2011)
19. Terefe, M.B., Lee, H., Heo, N., Fox, G.C., Oh, S.: Energy-efficient multisite offloading policy using markov decision process for mobile cloud computing. Pervasive Mob. Comput. **27**(C), 75–89 (Apr 2016). https://doi.org/10.1016/j.pmcj.2015.10.008, https://doi.org/10.1016/j.pmcj.2015.10.008
20. Xing, C., Chen, M., Yang, L.: Predicting available bandwidth of internet path with ultra metric space-based approaches. In: GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference. pp. 1–6 (2009)
21. Yu, Y.: Information Processing and Routing in Wireless Sensor Networks (2006)
22. Zeng, D., Gu, L., Yao, H.: Towards energy efficient service composition in green energy powered cyber–physical fog systems. Future Generation Computer Systems **105**, 757 – 765 (2020). https://doi.org/https://doi.org/10.1016/j.future.2018.01.060

A Appendix

A.1 Link quality estimation

Mirko: [*link simmetrico: banda e latenza*]

Communication latency and bandwidth are the two main metrics affecting energy consumption during data transfer between nodes [**?**]. Hence, having an estimate of how far, or what is the bandwidth, between a given service provider S and service consumer S', would allow assessing the energy consumption incurred for communication cost. **Mirko:** [*queste grandezze (e.g., $l_{s,s'}$ per la latenza e $b_{s,s'}$ per la banda) sono relazionate in qualche modo con $\kappa_S^{In}(S')$ e $\kappa_S^{Out}(S')$, in che modo?*] .

Mirko: [*il costo energetico di gossiping e di altre operazioni per fare stima di attributi di qualita' del link puo' essere considerato costante, uguale per tutti i nodi ed inglobato in qualche costante di costo energetico.*]

Mirko: [*nella sperimentazione definire classi di applicazioni su cui fare la sperimentazione. Determinate applicazioni sono ad esempio data intensive and latency sensitive (e.g., IoT), altre compute intensive (e.g., computational offload of heavy tasks). Queste classi potrebbero caratterizzare la scelta di alcuni parametri del modello nelle sperimentazioni*]

Latency Communication delay for interactions with services located at different nodes could have a non-negligible impact on the QoS and energy footprint of the system, as some services could be offered by distant fog/edge nodes. Due to the potential mobility of edge nodes, the communication delay may vary at runtime. This rules out approaches where these metrics are know in advance and the assembly problem solved at design-time as an optimization problem (e.g., [22]).

Estimating the latency among services located at different nodes would in principle require probing all pairwise link distances, which would not scale with high numbers of concrete services and hosting nodes. For this reason, similarly to [4], we adopt a network coordinates (NC) system that provides an accurate estimate of the latency between any two network locations, without the need of an exhaustive probing. The nodes maintain the NC system through a decentralized algorithm [5] with linear complexity with respect to the number of network locations, thus ensuring scalability. Moreover, as this NC algorithm adopts a gossip-based information dissemination scheme, the components operations are easily integrated with the overall approach.

Bandwidth With the increasing advent of data streaming application, in addition to latency, bandwidth has emerged as a crucial factor characterising the quality of communication links. CDNs and edge computing applications have a new need to select servers based on bandwidth in addition to latency [14]. In fact, end-to-end path available bandwidth is an important metric for the performance optimisation in many high throughput distributed applications, such as

video streaming and file sharing networks [2]. Several solutions for bandwidth prediction exists in the literature.

The Sequoia [14] algorithm uses a tree-embedding model for bandwidth prediction. Sequoia maintains a collection of virtual trees between the participating hosts and provides efficient latency/bandwidth prediction. Inspired by Sequoia, Song et. al. [18] design a tree-based decentralized bandwidth prediction model that, differently from the previous work, does not require a centralized component that the other nodes must communicate with. PathGuru [20] predict outgoing and incoming bandwidth using measurement data to and from landmarks. Similarly to Sequoia a centralized component or an overlay of (landmark) superpeers is necessary to run the algorithm.

DMF [11] introduces a solution based on decentralized matrix factorization, proposed in the context of latency estimation, it can be used also for bandwidth estimation. **Mirko:** [*DMF claims to be better than Vivaldi as it does not suffer from the triangle inequality violations problem*] Similarly, Vivaldi algorithm can also be used for bandwidth prediction as there are no assumption on the elements characterizing the distance in the euclidean distance model. In this case, differently from latency, for bandwidth prediction Vivaldi algorithm is fed with the inverse of the available (bandwidth) measurements. Vivaldi and Sequoia heuristics produce symmetric bandwidth estimations.

Beaumont et. al. [2] perform available bandwidth prediction with the last-mile model, in which each node is characterised by its incoming and outgoing capacities. Their heuristic compute in a decentralized way the capacities of each node.

Mirko: [*It would be nice to use Vivaldi also for bandwidth estimation and integrate in the message exchange for latencies also the bandwidth value, however results from [2] shows that Vivaldi is not appropriate for bandwidth estimation as it exhibits higher prediction error with respect to other solutions. Can we adopt it anyway?*]

Mirko: [*Good solutions for us:*

- *The last-mile model [2]: (i) heuristic for a fully decentralized system, (ii) asymmetric incoming/outgoing bandwidth model could go well with our model.*
- *The work in. [18] is interesting, works in fully decentralized systems but only for symmetric bandwidth estimation. This guy did the Ph.D in decentralized network bandwidth prediction. He has different approaches already implemented in PeerSim but i can't find the code. He states in his dissertation (on drive): "our approach would be able to cover more varied data sets with high accuracy than last-mile. In addition, last-mile needs high communication costs coming from multiple convergence steps just like DMF."*

]

A.2 Communication Energy Consumption Models

Communication energy consumption could be affected by many factors. Three of the main elements used in the literature to model this aspect are: latency (or distance), bandwidth and load (rate of data/service requests).

Mirko: [*we could have different communication energy models for different deployments/applications. Having multiple models allows us to easily refer to enstablished energy models in the literature.*

- *Our work define the communication energy consumption in relation to* $\kappa_S^{In}(S')$ *and* $\kappa_S^{Out}(S')$
- *A concrete deployment (e.g., wsn) lead to the definition of* $\kappa_S^{In}(S')$ *and* $\kappa_S^{Out}(S')$ *in relation to the performance quality attributes of the link (i.e.,latency and bandwidth, more?)*
- *What to do in case of mixed deployment of our framework (e.g., cloud-fog-edge)? The energy depends on type of node and link used? (in this case every node has its own definition of kappas).*

]

Latency-centric model In Wireless Sensor Network (WSN) the total energy is restricted and how to make good use of the limited energy resource is a very important aspect. For WSN applications a latency-centric communication energy model can be adopted. In this scenario, a commonly used communication consumption model in the literature is the first-order radio model [7]. This model ignores the effects of finite bandwidth constraints and models the communication energy consumption in relation to the latency metric. **Mirko:** [*this model is originally build on distance. But distance is directly related to latency*]

We define as constants for each node: E_{amp}, i.e., the amplification cost factor and E_{elect}, i.e., circuit energy cost when transmitting or receiving one bit of data. Considering a single (k-bit) interaction between S and S' with a latency, $l_{S,S'}$, the transmission energy cost can be expressed as:

$$\kappa_S^{In}(S') = k \cdot (E_{elect} + E_{amp} \cdot l_{S,S'}^2) \tag{25}$$

Likewise, the energy consumed at reception:

$$\kappa_{S'}^{Out}(S) = k \cdot E_{elect} \tag{26}$$

Bandwidth-centring model In data-intensive applications where, differently from WSN, large data volumes are moved between sites over the network, ignoring the effects of finite bandwidth constraints on communication energy consumption is not an option. In fact, if the throughput is low, the total time spent on the communication channel to send the data is longer, this results in higher energy consumption.

In this scenario, the communication time depends on the bytes of data transferred and the network throughput between the nodes hosting the services. Considering a single (k-bit) interaction between S and S' with a throughput, $b_{S,S'}$, the communication time spent on the channel can be expressed as:

$$t_{S,S'} = \frac{k}{b_{S,S'}} \tag{27}$$

Thus, similarly to [19] the transmission energy can be expressed as:

$$\kappa_S^{In}(S') = p_s \cdot t_{S,S'} \tag{28}$$

Likewise, the energy consumed at reception:

$$\kappa_{S'}^{Out}(S) = p_r \cdot t_{S,S'} \tag{29}$$

Where p_s and p_r are the power consumption when sending and receiving data, respectively. Based on measured results $p_s > p_r$ [10].