SensorScope, un système clef en main de surveillance de l'environnement

François Ingelrest — Guillermo Barrenetxea — Martin Vetterli

Laboratoire de Communications Audiovisuelles (LCAV)
École Polytechnique Fédérale de Lausanne (EPFL)
Station 14, EPFL, 1015 Lausanne, Suisse

{Francois.Ingelrest, Guillermo.Barrenetxea, Martin.Vetterli}@epfl.ch

RÉSUMÉ. *Étant donné l'importance des changements climatiques et leurs conséquences potentiellement dramatiques pour l'Homme, la mise au point de modèles d'évolution précis et fiables est d'une importance capitale. Malheureusement, les méthodes actuelles de collecte de données sont plutôt limitées car elles se basent sur des stations peu maniables et très chères, conduisant à un cruel manque d'observations spatiales et temporelles suffisamment denses. Les réseaux de capteurs représentent une alternative parfaitement adaptée pour répondre à ce problème, et peuvent donc avoir un impact considérable dans ce domaine. Nous présentons ici SensorScope, un projet multidisciplinaire dont le but est de fournir « clef en main » un tel système de surveillance, depuis les stations elles-mêmes jusqu'à l'infrastructure de gestion des données. Nous nous concentrons sur la collecte de ces dernières, et plus particulièrement sur les protocoles de communication qui la rendent possible. Nous détaillons également l'un de nos déploiements, effectué durant deux mois dans un glacier rocheux en Suisse, qui a permis la modélisation d'un micro-climat jouant un rôle important dans de dangereuses coulées de boue.*

ABSTRACT. *Given the importance of climate changes and their potentially dramatic consequences for human beings, designing precise and reliable evolution models is of prime importance. Unfortunately, current data collection techniques are rather limited as they make use of huge and very expensive sensing stations, leading to a lack of dense spatial and temporal observations. Wireless sensor networks are an alternative solution well-fitted to this problem, and may as such have a considerable impact on this domain. In this paper, we present SensorScope, a multidisciplinary project that aims at providing such an "out-of-the-box" monitoring system, from the sensing stations to the data management infrastructure. Here we especially focus on data gathering and on the communication protocols used for this task. We also describe one of our deployments, carried out during two months on a rock glacier in Switzerland, which led to the modeling of a micro climate that plays an important role in dangerous mud streams.*

MOTS-CLÉS : *Réseaux de capteurs, surveillance de l'environnement, déploiements.*

KEYWORDS: *Sensor networks, environmental monitoring, deployments.*

1. Introduction

Les réseaux de capteurs sont des réseaux sans fil, multi-sauts et auto-organisés, composés de *motes* déployés sur une zone à surveiller. Ces motes sont de petits objets, équipés d'un processeur et d'une radio, capables de relever diverses informations sur leur environnement par le biais de capteurs intégrés ou externes. Leur consommation énergétique est souvent une préoccupation majeure, étant donné le caractère éloigné et/ou inaccessible des zones surveillées qui rend difficile le remplacement des batteries. Les contraintes qui en résultent expliquent la présence de stations plus puissantes (des *puits*) au sein du réseau, disposant d'un moyen de communication longue portée, comme un GPRS, permettant l'envoi des données recueillies à un serveur distant.

Bien que ces réseaux possèdent de nombreuses applications, la surveillance de l'environnement est l'un des domaines où leur apport peut avoir un impact très important. Les dramatiques accidents climatiques que nous avons récemment connus ont en effet montré à quel point la connaissance de notre environnement et de son évolution peut être cruciale pour l'Homme. Toutefois, la capacité des chercheurs à améliorer celle-ci est restreinte par les méthodes actuelles de collecte des données, basées sur des stations très chères et disposant d'une capacité de stockage limitée.

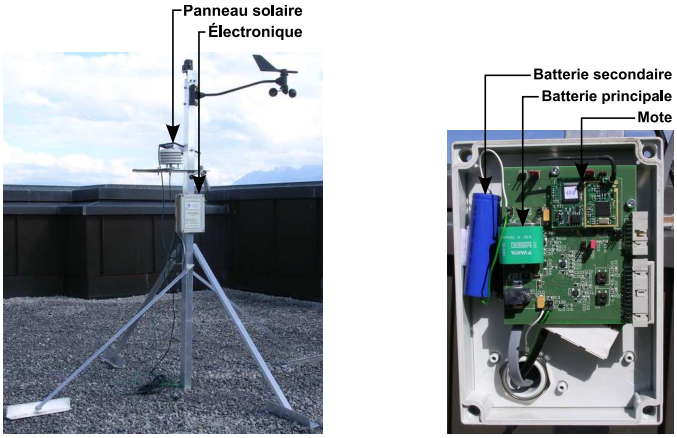
Dans cet article, nous présentons SensorScope, un système de surveillance de l'environnement. Parce qu'il repose sur un réseau de capteurs, il lève les limitations actuelles, notamment grâce à son aptitude à pouvoir transmettre immédiatement les données recueillies vers un serveur capable de les stocker durablement et en grande quantité. Celles-ci sont alors rendues publiquement disponibles en temps réel sur notre site Internet. En rassemblant les différents résultats de recherche sur les réseaux de capteurs, nous avons mis au point une pile de communication complète basée sur TinyOS, librement téléchargeable et utilisable sous une licence open-source[1]. Sa fiabilité et sa robustesse ont été largement éprouvées durant l'automne 2007, au cours duquel nous avons déployé plusieurs réseaux dans différents lieux en Suisse, dont l'un situé à 2 500 m d'altitude dans la montagne du Génépi durant deux mois.

Dans la section suivante, nous survolons les architectures matérielle et réseau de SensorScope, puis nous détaillons en section 3 les caractéristiques majeures de notre pile de communication. Dans la section 4, nous présentons les résultats obtenus durant le déploiement du Génépi cité plus tôt, puis nous concluons finalement en section 5.

2. Architecture du réseau de capteurs

Comme cela a été mentionné précédemment, le manque de relevés continus sur notre environnement bride notre capacité à le modéliser et donc à prédire son évolution. Les campagnes actuellement menées reposent sur un petit nombre de stations très chères, limitant la couverture spatiale des observations. De plus, ces stations stockent elles-mêmes les données recueillies dans une mémoire d'une taille limitée. Un réseau

1. `http://sensorscope.epfl.ch/network_code`



(a) Aperçu de la station. (b) Électronique.

Figure 1. *Une station de collecte SensorScope.*

de capteurs représente la solution idéale à ces problèmes, et permet une surveillance aussi bien en temps réel (pollution) qu'à long terme (fonte des glaces) d'événements naturels au sein de zones de taille plus ou moins grande.

Partant de cette idée, nous avons développé SensorScope, un système de surveillance de l'environnement basé sur un tel réseau. Notre but est qu'il puisse être livré « clef en main », c'est-à-dire nécessitant une connaissance et une maintenance minimales du réseau, de manière à ce qu'il puisse être utilisé aisément par des chercheurs en environnement sans qu'une armada d'informaticiens ne soit présente sur les lieux. Notre interaction au sein du projet a, de ce point de vue, été essentielle.

2.1. *Architecture matérielle*

Le mote que nous avons sélectionné est un TinyNode[2], composé d'un processeur 16 bits MSP430 cadencé à 8 MHz et d'une radio Semtech XE1205 opérant dans la bande des 868 MHz à un débit de 76 Kbps. Il dispose de 48 Ko de ROM et de 10 Ko de RAM. Nous l'avons sélectionné car il possède une grande portée, de l'ordre de 200 m en extérieur, pour une consommation électrique assez basse [DUB 06].

Quant à la station elle-même, représentée en figure 1a, elle repose sur un squelette en aluminium sur lequel sont fixés un panneau solaire et les capteurs. Sa taille (1,5 m) et ses quatre pieds la rendent stable et lui permettent de supporter les chutes de neige hivernales. La figure 1b montre que les parties électroniques sont enfermées dans une boîte hermétique placée juste au-dessus des pieds. Le prix total d'une station, incluant le mote, le panneau solaire et les capteurs, tourne autour de 900 €.

2. `http://www.tinynode.com`

2.1.1. *Alimentation électrique*

Afin d'obtenir une autonomie suffisante lors des déploiements, chaque station est équipée d'un module de gestion d'énergie solaire formé de trois composants :

Panneau solaire D'une taille de 162×140 mm et d'une durée de vie de vingt ans, il fournit une puissance maximale de 1 W en éclairage direct.

Batterie principale D'une capacité de 150 mAh et de type NiMH, elle est la source d'alimentation principale et est rechargée par le panneau solaire.

Batterie secondaire Cette batterie Li-Ion de 2 200 mAh alimente la station lorsque la capacité de la première tombe en-dessous d'un certain seuil. Elle est rechargée par la batterie principale lorsque celle-ci est à nouveau pleine.

Ce système, avec les algorithmes d'économie d'énergie de la pile de communication, rend virtuellement infinie la durée de vie d'une station. Nous avons en effet pu constater durant nos déploiements que les batteries restent toujours chargées même en cas de temps nuageux, comme nous le montrons plus loin dans la section 4.

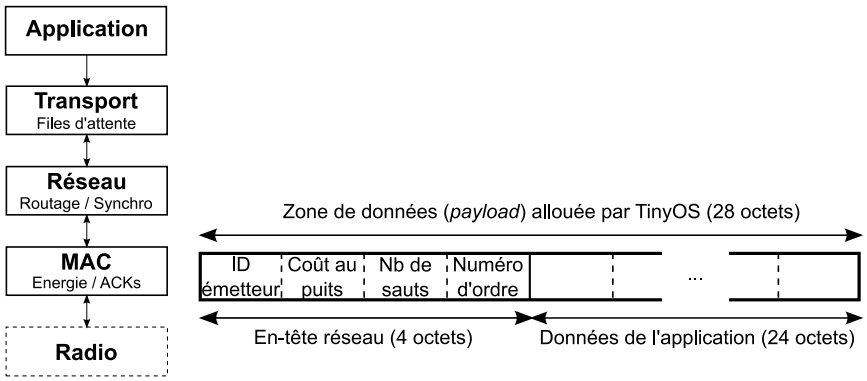
2.1.2. *Capture de données*

Jusqu'à sept capteurs peuvent être installés sur chaque station. Ceux que nous avons choisis nous permettent de mesurer neuf quantités : température de l'air et du sol, humidité de l'air, taux d'absorption et humidité du sol, vitesse et direction du vent, radiation solaire et précipitations. Afin de nous assurer de la qualité des mesures, les capteurs sont calibrés durant plusieurs jours avant d'être déployés en comparant leurs mesures à des stations de référence. Nous exigeons un coefficient de corrélation d'au moins 0.98 pour valider un capteur.

2.2. *Architecture réseau*

Pour prendre en charge le transfert des données depuis les stations jusqu'au puits, nous avons choisi de concevoir et de développer notre propre pile de communication. Nous avons fait ce choix car il est parfois très difficile de bien comprendre les interactions entre des composants existants écrits séparément, de façon à les assembler de manière correcte. De plus, ces efforts n'en valent parfois pas la peine, par exemple lorsque ces composants sont très complexes du fait de fonctionnalités qui ne sont pas nécessaires pour l'application visée.

Notre pile de communication, illustrée par la figure 2a, est implémentée sous TinyOS et suit la structure du modèle OSI. Elle n'a besoin que de quatre octets pour l'en-tête réseau, stockée dans la zone de données des paquets, ce qui laisse 24 octets pour l'application sur les 28 disponibles comme le montre la figure 2b. Nous avons choisi cette méthode, plutôt que de modifier l'en-tête standard utilisée par TinyOS, afin d'être indépendants de la radio et de ses pilotes. Il y a quatre couches différentes :



(a) La pile. (b) Format d'un paquet.

Figure 2. *La pile de communication développée pour SensorScope.*

Application Cette couche est responsable de la collecte des données (dont le niveau d'énergie des batteries) à envoyer au puits.

Transport Elle crée et reçoit les paquets réseau, puis les stocke si besoin dans une file d'attente selon leur type : les paquets de données (comme les mesures) sont routés vers le puits, ceux de contrôle (comme les ACKs) sont envoyés à un voisin en particulier. Le trafic étant faible, il n'y a pas besoin de mécanisme de contrôle de congestion. Deux champs de l'en-tête y sont également remplis : le nombre de sauts effectués par le paquet et le numéro d'ordre.

Réseau Elle passe les paquets à la couche MAC après avoir choisi un prochain saut pour ceux de données. Elle remplit les champs contenant l'identifiant de l'expéditeur et le coût de la route jusqu'au puits. Toutes les décisions de routage sont prises à ce niveau, et implémenter un nouveau protocole ne nécessite que l'écriture d'une nouvelle couche réseau. Le protocole utilisé dans SensorScope ainsi que la valeur des champs sont détaillés dans la prochaine section.

MAC Afin de réduire la consommation énergétique, elle éteint la radio dès que possible. Les paquets ne sont bien sûr envoyés que lorsque cette dernière est allumée, le reste du temps ils sont conservés et transmis plus tard. Seuls les paquets de données sont acquittés par un ACK, renvoyé au précédent saut. Nos pilotes radio ne possédant pas de mécanisme de détection d'activité du canal radio, un simple mécanisme d'attente aléatoire est utilisé pour minimiser les collisions.

3. Principales fonctionnalités de la pile de communication

Dans ce qui suit, la distance est toujours exprimée en nombre de sauts et ne désigne pas la distance euclidienne. De même, le terme *diffusion* désigne un paquet envoyé à tous les voisins directs de l'expéditeur et non pas à toutes les stations du réseau.

3.1. *Détection du voisinage réseau*

Chaque mote maintient une table de voisinage dans laquelle il stocke les voisins dont il reçoit les paquets. Nous avons choisi une méthode par *overhearing* dans l'esprit de MintRoute [WOO 03], c'est-à-dire qu'il n'y a pas de paquets dédiés mais que les motes découvrent leur voisinage grâce au trafic de données, et seul le puits doit initier le processus avec de véritables messages de signalisation. Chaque entrée de la table est associée à un coût, ce dernier étant actuellement la distance vers le puits, et une date de mise à jour. Les voisins n'étant plus assez à jour sont automatiquement supprimés.

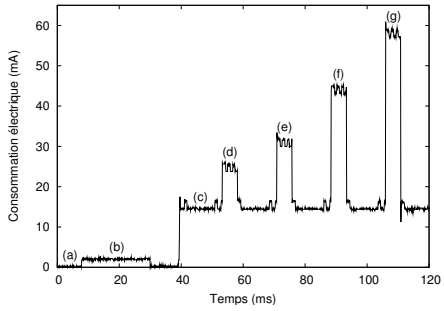
Pour éviter d'utiliser des voisins trop éloignés, une qualité de lien (QoS) est calculée pour chacun en comptant les « trous » dans les numéros d'ordre des paquets : plus les pertes sont nombreuses et plus la QoS chute. Une alternative que nous avons considérée est l'utilisation de la puissance du signal reçu, mais cette dernière varie énormément à chaque réception, et ne donne pas de bons résultats. Une remarque importante est que seuls les paquets de données sont utilisés pour le calcul de la QoS. Ainsi, puisque ces paquets sont renvoyés avec le même numéro d'ordre lorsqu'ils ne sont pas acquittés, ils font baisser la QoS des voisins qui n'arrivent pas à faire suivre les données vers le puits. La qualité reflète donc au final non seulement la capacité du voisin à être entendu, mais aussi sa capacité à router efficacement les paquets.

3.2. *Synchronisation sur une heure de référence*

Pour diverses raisons, comme une perte temporaire du signal radio, les paquets peuvent rester bloqués quelques temps à une station, ces délais empêchant le serveur de dater lui-même les données. Il est donc nécessaire que les stations possèdent une heure de référence, que nous appelons *heure réseau*, utilisée pour dater les données recueillies. Le cristal des motes étant sujet à une dérive temporelle, il est indispensable que cette heure soit régulièrement mise à jour : la dérive d'un TinyNode est par exemple de ± 20 ppm, soit une seconde toutes les 14 heures.

Dans SensorScope, des messages `SYNC_REQUEST`/`SYNC_REPLY` sont utilisés, le but étant de propager l'heure locale du puits, arbitrairement désignée comme l'heure réseau. Quand une station veut mettre à jour son horloge, elle choisit un voisin *plus proche qu'elle* du puits et lui envoie une requête. Ce dernier diffuse alors sa réponse à ses voisins qui mettent tous à jour leur heure s'ils sont *plus éloignés* du puits que lui. En procédant ainsi, l'heure se propage toujours depuis le puits qui reste la référence pour tout le réseau, et ce même si sa propre horloge dérive. Le puits envoie également régulièrement son heure au serveur qui, connaissant l'heure absolue, peut calculer le décalage et ajuster la date des données reçues afin de leur affecter une heure absolue.

Les requêtes de synchronisation sont gérées par la couche réseau, mais c'est la couche MAC qui met à jour l'heure contenue dans les paquets, éliminant ainsi les erreurs de délais [GAN 03]. Deux modes sont utilisés : un mode haute fréquence, employé lorsque l'heure réseau n'est pas connue, et un mode basse fréquence util-



Zone	Action	Valeur
(a)	Processeur inactif	0 mA
(b)	Processeur actif	2 mA
(c)	Radio allumée	15 mA
(d)	Transmission (0 dB)	25 mA
(e)	Transmission (5 dB)	31 mA
(f)	Transmission (10 dB)	44 mA
(g)	Transmission (15 dB)	58 mA

Figure 3. *Consommation électrique d'un TinyNode selon son activité.*

isé le reste du temps. D'après notre expérience, une mise à jour toutes les heures pour ce dernier mode fournit une précision d'environ 70 millisecondes, suffisante pour nos applications. Il existe des protocoles fournissant une meilleure précision, mais apportant une complexité non négligeable qui peut s'avérer inutile. Par exemple, FTSP [MAR 04] tente de compenser la dérive temporelle par des calculs de régression linéaire, calculs qui sont fortement liés au mote choisi et à sa dérive, alors que celle-ci peut considérablement varier avec les conditions climatiques telle que la température.

3.3. *Gestion de la radio*

Une bonne gestion de l'énergie est essentielle pour des déploiements de longue durée, et bien que notre système à base d'énergie solaire soit efficace, la consommation de la radio du mote est suffisamment élevée pour être prise en considération, comme le montre la figure 3. Même à l'état de « veille », c'est-à-dire uniquement à l'écoute du canal, la radio une fois allumée multiplie par sept la consommation électrique, et peut par conséquent rapidement mener à une balance énergétique négative.

Il est donc indispensable que les motes s'organisent en cycles de communication à deux états : un état *actif*, dans lequel la radio est allumée et où les paquets sont envoyés/reçus, et un état de *veille* où la radio est éteinte. L'obtention de bonnes économies d'énergie nécessite évidemment que ce dernier état soit le plus long possible. Cette organisation peut s'effectuer de deux manières :

Asynchrone Cette solution nécessite qu'un *préambule* (une suite de bits spécifique) suffisamment long soit envoyé avant chaque paquet, de manière à ce que les nœuds proches le détectent et restent éveillés en attente du paquet. B-MAC utilise ce procédé [POL 04].

Synchrone À l'opposé, cette méthode nécessite que tous les nœuds allument leur radio en même temps, ce qui permet de se passer d'un préambule. TASK utilise un tel mécanisme [BUO 05].

Nous avons choisi la seconde solution, car la première augmente de manière sensible la durée des transmissions, le préambule devant être plus long que l'état de veille, et lorsque le trafic n'est pas assez faible cela peut entraîner des congestions. Il a également été démontré [BUO 05] que la première solution entraîne une consommation énergétique un peu plus élevée et réduit donc la durée de vie totale.

Réveiller les stations simultanément est assez simple dans SensorScope, grâce au mécanisme de synchronisation présenté précédemment. Les nœuds ne connaissant pas encore l'heure réseau gardent simplement leur radio allumée, en attendant d'être synchronisés. Afin de prendre en compte une légère dérive temporelle, les stations attendent quelques dizaines de millisecondes au début de leur état actif sans rien envoyer, pour s'assurer que leurs voisins sont bien eux-mêmes éveillés. TinyOS fonctionnant par événements, cette division en cycles de communication est transparente pour les couches supérieures : si un paquet doit être envoyé alors que la radio est éteinte, il est conservé en attendant le prochain cycle et la couche du dessus attend simplement le signal `sendDone`, la durée de cette attente n'ayant aucun impact.

3.4. *Routage des données vers le puits*

Pour router les données, une solution généralement considérée est de créer puis de maintenir une structure en forme d'arbre dont la racine est le puits. Nous avons choisi de ne pas utiliser cette méthode, car elle implique un coût de maintenance, pour détecter les déconnections, mais également pour parvenir à un bon équilibre de la charge de routage. On peut en effet aisément imaginer qu'une station devienne le prochain saut de plusieurs autres, entraînant sa surcharge non seulement à cause de ces stations, mais aussi à cause de celles se situant plus bas dans l'arbre. Une telle situation peut par exemple se produire lorsque les nœuds sont reliés à leur meilleur parent, quelle que soit la métrique considérée, comme c'est le cas dans MintRoute [WOO 03].

Pour éviter ces problèmes, lorsqu'une station doit router des données, elle choisit toujours son prochain saut au hasard, rendant en quelque sorte le routage *opportuniste* avec une grande diversité de routes. Afin de s'assurer que les données finiront toujours par arriver, l'ensemble des prochains sauts est constitué des voisins plus proches du puits que la station elle-même. Puisqu'à chaque étape le choix est aléatoire, l'équilibre de la charge s'effectue automatiquement et chaque nœud est sollicité de la manière la plus équitable possible. Afin de valoriser les meilleurs voisins, nous avons défini deux seuils, bonne et mauvaise QoS : quand un paquet doit être routé, un voisin de bonne qualité est pris aléatoirement et s'il n'y en a pas, alors un voisin de mauvaise qualité est choisi. Les voisins en-dessous de la mauvaise qualité ne sont pas pris en compte.

4. Résultats de déploiements

Au cours de l'année 2007, nous avons effectué six déploiements en extérieur, depuis le campus de l'EPFL jusqu'à des zones de haute montagne. Nous détaillons

(a) Vue globale. (b) Vue locale.

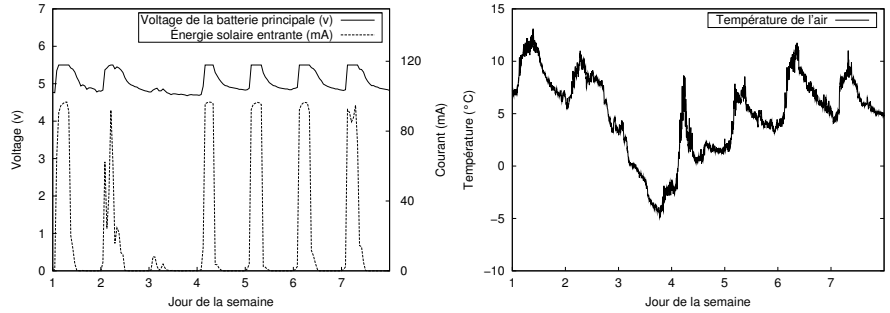
Figure 4. *Carte du déploiement du Génépi et position des stations.*

Couche	Paramètre	Valeur
Application	Échantillonnage	120 s
Réseau	Timeout d'un voisin	480 s
	Voisin de bonne qualité	$\geq 90\,\%$
	Voisin de mauvaise qualité	$\geq 70\,\%$
	Haute fréquence de synchronisation	5 s
	Basse fréquence de synchronisation	1 h ($\pm$ 72 ms de dérive)
Mac	Puissance d'émission	15 dB
	Durée de l'état actif	12 s
	Durée de l'état de veille	108 s

Tableau 1. *Paramètres utilisés durant la campagne du Génépi.*

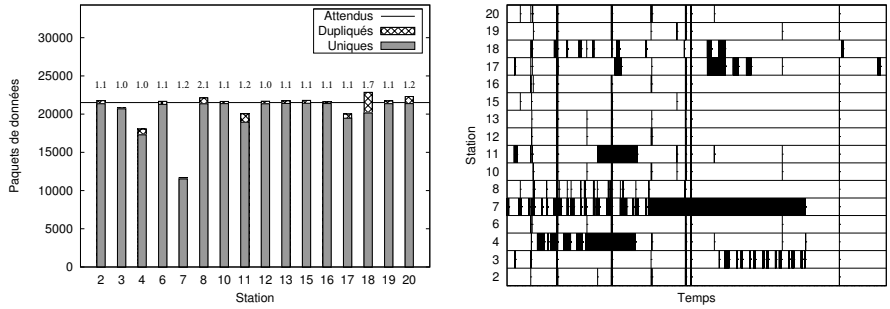
ici l'un de nos plus importants déploiements, réalisé dans un glacier rocheux situé à 2 500 m d'altitude sur la montagne du Génépi en Suisse. Ce site a été choisi car, lors de pluies intenses, il est la source de coulées de boue ayant tué plusieurs personnes ces dernières années. Les autorités cantonales en charge de la sécurité n'ayant à ce moment aucun modèle climatique du site, elles nous ont demandé d'y déployer un réseau afin de corréler les précipitations avec le vent et la température en fonction de la forme du terrain. Elle nous ont fourni toute l'aide technique pour cela, incluant un hélicoptère pour le transport du matériel et des personnes. Ce déploiement a été effectué durant les derniers jours du mois d'août 2007 et a duré plus de deux mois.

Les 16 stations ont été déployées sur une zone de 500×500 m comme le montre la figure 4. Leur position a été choisie avec attention, afin d'obtenir des observations significatives : la station 20 a ainsi été placée dans la zone de rupture du glacier et la 11 dans la pente de rochers. Afin de transmettre les données à nos serveurs, le puits, situé près de la station 3, était équipé d'un module GPRS que nous utilisions pour la première fois. Comme nous le montrons plus loin, son fonctionnement a été problématique et a causé la perte de quelques données, notamment à cause de la faible connectivité sur le site. Les différents paramètres utilisés sont fournis par le tableau 1.



(a) Énergie disponible. (b) Température de l'air observée.

Figure 5. *Une semaine dans le système d'énergie solaire d'une station.*

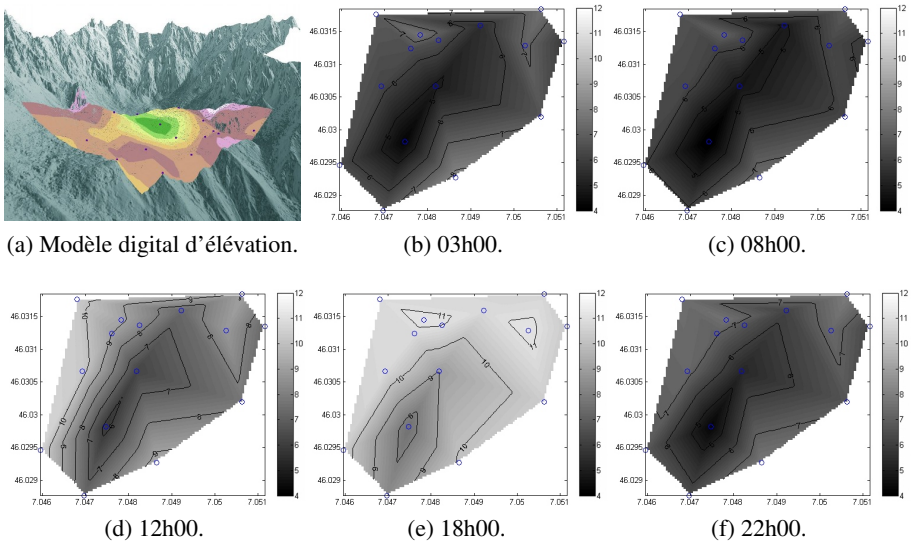


(a) Paquets de données reçus et distance moyenne des stations au puits. (b) Corrélation temporelle des pertes.

Figure 6. *Statistiques des données reçues durant un mois complet.*

Cette campagne a été une bonne occasion de tester l'autonomie des stations en conditions réelles. La figure 5 montre ainsi la variation d'énergie de la station 15 pendant une semaine complète, l'énergie solaire recueillie ainsi que la corrélation avec la température observée par cette même station. Les mesures commencent le matin du 16 septembre 2007 à 6h00, période durant laquelle le lever du soleil était autour de 6h00 et son coucher autour de 21h00, et montrent la consommation totale de la station, incluant ses capteurs. On peut nettement voir que la batterie principale se vide lentement la nuit puis se recharge complètement la journée. Au troisième jour, le temps a été très nuageux, impliquant une chute brutale de la température, mais malgré le faible ensoleillement la charge fut suffisante. Durant toute la semaine, la batterie secondaire n'a pas du tout été sollicitée, ce qui prouve l'efficacité du système.

La figure 6 donne les statistiques de la collecte des données durant un mois à partir du 10 septembre 2007, et montre que la plupart des paquets a été recueillie pour la



(a) Modèle digital d'élévation. (b) 03h00. (c) 08h00.

(d) 12h00. (e) 18h00. (f) 22h00.

Figure 7. *Modèle d'élévation (0.5×0.5 m) et distribution spatiale de la température de l'air le 2 octobre 2007.*

majeure partie des stations. Certaines d'entre elles, notamment les stations 4, 7, 11 et 17, ont été victimes de problèmes matériels, tels que des court-circuits, qui ont entraîné la perte d'un certain nombre de paquets consécutifs. Ce phénomène est très visible sur la figure 6b où l'on peut voir la corrélation temporelle des pertes par station, chaque trait représentant un paquet manquant. Cette figure met également en avant les problèmes liés au GPRS, ces derniers se manifestant par des pertes simultanées pour toutes les stations. Nous savons à présent qu'elles ont été causées par un bug dans le programme du GPRS, entraînant la suppression des paquets mémorisés lors d'une perte de la connexion au réseau cellulaire.

Finalement, la figure 7 fournit une partie des résultats de l'analyse actuellement menée par nos collaborateurs en mécanique des fluides de l'environnement. Elle montre la position des stations sur le modèle d'élévation digitale du glacier rocheux, au centre duquel on peut voir la vallée où le permafrost est le plus épais (jusqu'à 15 m) et où les coulées de boue prennent leur source. Les autres figures montrent la distribution spatiale de la température de l'air le 2 octobre 2007, journée durant laquelle le ciel était parfaitement dégagé. Chaque valeur est la moyenne d'une heure de mesures.

Ces résultats montrent que même en cas de ciel ensoleillé, la température est toujours basse dans la vallée. En effet, tandis que la variation de température est autour de 5°C à la bordure du site, elle n'est que de 2°C dans la vallée. Cette observation est intéressante car toutes les stations étaient exposées de la même manière au soleil, et auraient donc dû reporter les mêmes valeurs. Cette différence s'explique par l'influence de l'épaisse couche de glace du permafrost, conservant la température à une

valeur assez basse quelle que soit l'exposition au soleil. Ce micro-climat est actuellement considéré comme l'un des éléments clefs dans la prédiction des coulées de boue.

5. Conclusion

À travers les déploiements effectués, SensorScope est devenu un projet clef, fusionnant les technologies des réseaux de capteurs et celles de la recherche environnementale. La campagne du Génépi a été très enrichissante, non seulement en termes de données collectées mais également sur le plan de l'expérience acquise. Les défauts de jeunesse auxquels nous avons fait face, notamment liés au matériel, sont en cours de résolution et ne devraient plus poser de problème à l'avenir. L'implémentation d'un système de reprogrammation à distance tel que Deluge [HUI 04] est également à l'étude, afin de minimiser les déplacements sur le terrain une fois le réseau déployé.

Remerciements

Ces travaux ont été en partie financés par NCCR-MICS, par le projet Européen FP 6 WASP, ainsi que par Microsoft Research.

6. Bibliographie

[BUO 05] BUONADONNA P., GAY D., HELLERSTEIN J., HONG W., MADDEN S., « TASK : Sensor Network in a Box », *Proceedings of the IEEE European Workshop on Wireless Sensor Networks and Applications (EWSN)*, janvier 2005.

[DUB 06] DUBOIS-FERRIÈRE H., MEIER R., FABRE L., METRAILLER P., « TinyNode : A Comprehensive Platform for Wireless Sensor Network Applications », *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, avril 2006.

[GAN 03] GANERIWAL S., KUMAR R., SRIVASTAVA M., « Timing-Sync Protocol for Sensor Networks », *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, novembre 2003.

[HUI 04] HUI J., CULLER D., « The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at a Scale », *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, novembre 2004.

[MAR 04] MARÓTI M., KUSY B., SIMON G., LÉDECZI A., « The Flooding Time Synchronization Protocol », *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, novembre 2004.

[POL 04] POLASTRE J., HILL J., CULLER D., « Versatile Low Power Media Access for Wireless Sensor Networks », *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, novembre 2004.

[WOO 03] WOO A., TONG T., CULLER D., « Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks », *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, novembre 2003.