Atmos. Meas. Tech., 9, 991–999, 2016
www.atmos-meas-tech.net/9/991/2016/
doi:10.5194/amt-9-991-2016

Atmospheric Measurement Techniques
Open Access

Real-time data acquisition of commercial microwave link networks for hydrometeorological applications

Christian Chwala[1], **Felix Keis**[1], **and Harald Kunstmann**[1,2]

[1]Karlsruhe Institute of Technology (KIT), Institute of Meteorology and Climate Research (IMK-IFU), Garmisch-Partenkirchen, Germany
[2]University of Augsburg, Institute for Geography, Augsburg, Germany

Correspondence to: Christian Chwala (christian.chwala@kit.edu)

Received: 11 September 2015 – Published in Atmos. Meas. Tech. Discuss.: 23 November 2015
Revised: 18 February 2016 – Accepted: 22 February 2016 – Published: 9 March 2016

Abstract. The usage of data from commercial microwave link (CML) networks for scientific purposes is becoming increasingly popular, in particular for rain rate estimation. However, data acquisition and availability is still a crucial problem and limits research possibilities. To overcome this issue, we have developed an open-source data acquisition system based on the Simple Network Management Protocol (SNMP). It is able to record transmitted and received signal levels of a large number of CMLs simultaneously with a temporal resolution of up to 1 s. We operate this system at Ericsson Germany, acquiring data from 450 CMLs with minutely real-time transfer to our database. Our data acquisition system is not limited to a particular CML hardware model or manufacturer, though. We demonstrate this by running the same system for CMLs of a different manufacturer, operated by an alpine ski resort in Germany. There, the data acquisition is running simultaneously for four CMLs with a temporal resolution of 1 s. We present an overview of our system, describe the details of the necessary SNMP requests and show results from its operational application.

1 Introduction

Only a decade ago, Messer et al. (2006) introduced the use of commercial microwave link (CML) networks for the quantification of rainfall. Since then, this technique has been applied further in several other countries. Leijnse et al. (2007) performed the first analysis of data from two CMLs in the Netherlands. Increasing the number of CMLs and using the data from countrywide networks, Zinevich et al. (2008) and Overeem et al. (2013) were able to derive spatial rainfall information in Israel and the Netherlands respectively. In Germany, Chwala et al. (2012) acquired the first data using data loggers at the CML towers. This limited the number of the CMLs but provided data with a very high power resolution.

Just recently, Doumounia et al. (2014) used CML data from a cell phone provider in Burkina Faso and showed that this technique is also feasible in West Africa. Due to the very coarse station network and the absence of weather radar, the additional CML-derived precipitation information is particularly important in African countries. Fortunately, the growing demand for mobile communication ensures an increasing number of CMLs also in this region.

Besides its application for the quantification of rainfall, CML data can also be used for the estimation of atmospheric humidity (David et al., 2009).

However, the availability of CML data remains one of the crucial hurdles for research and future operational applications at national meteorologic services and water authorities.

The software we have developed and which is running operationally in Germany strives to improve this situation. It is capable of acquiring data of homogeneous or heterogeneous CML networks at high temporal resolution in near-real time. In this article we describe the technical background and the details of the software, and we show results from its applications. Section 2 describes the problem of limited CML data availability for research. Section 3 introduces the concept of acquiring CML data via Simple Network Management Protocol (SNMP). Section 4 explains the details of the implementation of our real-time data acquisition system. Section 5 gives examples from two operational applications of the system. And Sect. 6 discusses the transferability to other CML networks.

With this development and initiative we finally aim not only to provide a useful tool for the hydrometeorological research with CMLs but also to encourage other CML operators to open their networks for custom data acquisition.

2 The problem: limited availability of CML data

2.1 Necessary data for rain rate estimation with CMLs

To be able to estimate rain rates with a microwave link, information of the attenuation along the path is necessary. This attenuation is strongly correlated with the rain rate along the path. To determine the attenuation, the transmitted (TX-level) and the received power (RX-level) of the links have to be known. For older CML hardware, the TX-level is typically constant, but modern systems support an adjustment of the TX-level via automatic transmission power control to compensate attenuation events and keep the RX-level in the receiver's optimal range. Hence, recordings of the RX-level, and very often also TX-levels, are necessary, ideally at a time resolution of minutes to capture the temporal variability of rainfall.

2.2 Situation for the CML operators

One of the biggest advantages of the hydrometeorological usage of CML networks is that they already exist, are well maintained and provide a good spatial coverage in inhabited areas (Overeem et al., 2013). The large countrywide CML networks are operated by cell phone providers or by CML manufacturers for a cell phone provider. Smaller networks are operated locally, e.g., by companies to establish wide area networks (WAN) between different remote sites.

For the operators of CML networks, the attenuation that is caused by rainfall is only unwanted disturbance, since it perturbs the communication along the CML in case of strong attenuation events. In periods with extremely heavy rainfall this may even lead to total loss of connection.

Strong attenuation events are not the only cause of failures in a CML network, though. Hardware and software problems occur and have to be fixed promptly to assure constant high availability of a network's full functionality. Hence, the CML networks are continuously monitored. But for the monitoring, neither the exact point in time, nor the exact amount of precipitation that may have caused a system failure is important.

As consequence the network operators store TX- and RX-level data only in a coarse resolution (15 min, hourly, daily) or do not store that data at all.

2.3 State-of-the-art CML data acquisition for research

For the purpose of monitoring, modern CML systems support the storage of a so-called performance report, which typically contains the TX-level and RX-level. Depending on the hardware type and the chosen settings these reports are generated every 15 min, every hour or daily. The most common setting is a temporal resolution of 15 min and the recording of the minimum and maximum values within this period. The TX- and RX-level resolution is typically 0.3 or 1 dB, depending on the hardware. These performance data are stored, e.g., by T-Mobile in the Netherlands (Overeem et al., 2013) and by cell phone providers in Israel (Zinevich et al., 2008) and are made available for research. Data transfer from the providers to the researches is usually made in chunks on daily or irregular basis.

In Germany the situation with our cooperation partner Ericsson is different. Ericsson operates the CML network for T-Mobile, but it does not store the CML performance reports. Hence, in our initial approach, we used small data loggers directly at the CML towers to record the RX-level, which is available at the analog output of the automatic gain control (AGC) of older hardware (Chwala et al., 2012). The TX-level was constant at the CMLs which we equipped. Data were sent to a database from the data loggers via GSM every day. This approach had the advantage of a very high power resolution of the RX-level, about 0.03 dB, only limited by the resolution of the module's analog-to-digital converter. However, it also had one main drawback. Equipping one link involved not only the costs for the data acquisition module (approximately Eur 100) but also the expenses for an Ericsson technician who did the actual installation at the towers. Hence, using dedicated data acquisition modules at the towers was not suitable for an extension to a countrywide coverage, involving several thousands of CMLs.

In joint cooperation with Ericsson Germany we thus developed a custom data acquisition solution which can easily be scaled up for a larger number of CMLs and which at the same time improves the temporal resolution compared to the available data from the performance reports.

Just recently, Fencl et al. (2015) have mentioned a similar approach, also using a custom data acquisition software running within an operational CML network. In contrast to the software that we are presenting here, which handles multiple parallel requests of CML data, their software is only capable of serial data acquisition. Hence, it seems not to be suitable to provide data for a large number of CMLs with a high and constant temporal resolution.

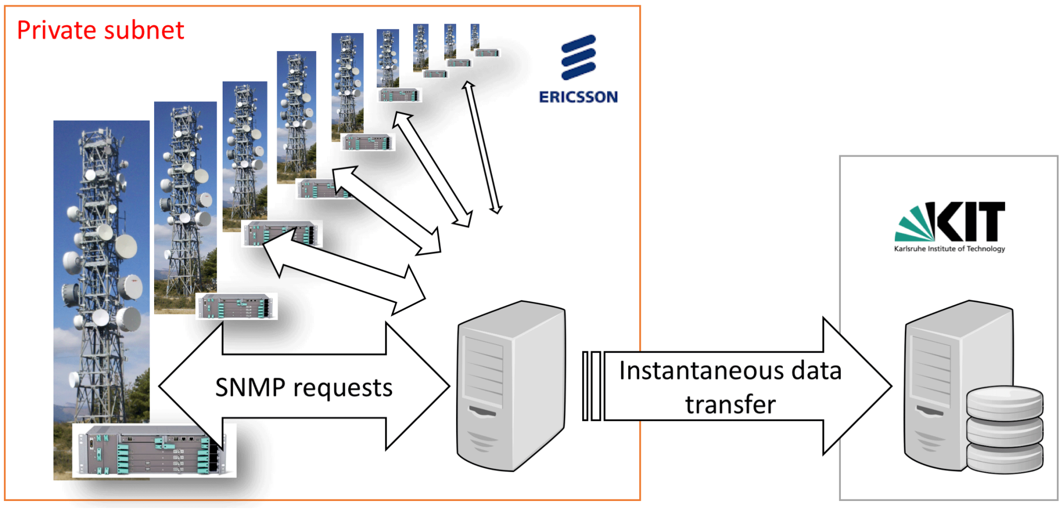
Private subnet
ERICSSON
SNMP requests
Instantaneous data transfer
KIT
Karlsruhe Institute of Technology

Figure 1. Overview of the basic concept of our real-time CML data acquisition system at Ericsson. The CMLs are shown together with one main control unit per site (this is a simplified example) which can be identified via its private IP address. The DAQ server inside the private subnet requests the CML signal levels via SNMP and immediately sends the data to a database server at our institute (KIT).

3 The solution: custom data acquisition via SNMP

3.1 The simple network management protocol (SNMP)

Since CMLs are built to form communication networks, it is easy to communicate with an individual CML when one has access to the network. Each CML is accessible via its IP address, which is typically from a private IP address space. For service and configuration related communication the Simple Network Management Protocol (SNMP) is used, which was developed to allow standardized communication between network attached devices, i.e., routers, printers or computers.

A SNMP request is defined by a target IP address and the SNMP object identifier (OID). The OID defines the actual command that has to be evoked. The OIDs are defined in the management information base (MIB). There are general MIBs which define OIDs that every system supports, like sysUpTime, which would request the uptime of the system. For additional functionality the MIBs can be extended. For example, genEquipRfuStatusTemperature is the OID to query the hardware temperature of the radio unit of a Ceragon IP10 CML.

3.2 SNMP for monitoring a CML

In large CML networks, like the ones operated by the cell phone providers, SNMP is used to remotely configure the CMLs and continuously monitor them for failures. Querying the TX- and RX-level is a typical functionality of the monitoring capabilities and is, to our knowledge, supported on all modern CMLs.

The OIDs to request the TX- and RX-level are not standardized and differ from CML hardware to CML hardware, even for the same manufacturer. For example, xfRFCurrentInputPower is the base OID for requests of the current output power for an Ericsson MINI-LINK Traffic Node CML while "gnOduStatusXReceiveLevel" is the one for a Ceragon IPMax and "genEquipRfuStatusRxLevel" for a Ceragon IP10 system. For each system they have to be looked up in the MIBs which are available to the CML owners from the manufacturers.

Since modern CML systems like the Ericsson MINI-LINK Traffic Nodes support the operation of several CMLs managed by one main control unit, the derivation of the correct OID requires a further step, though. The base OID has to be extended to specify which sub-system, hence which CML, has to be accessed. Furthermore, some modern systems make it possible to query data not only for the near end of a CML, to which the SNMP requests are sent, but also for the opposite site, the far end. This also has to be encoded in the OID.

Appendix A explains in detail how the correct OIDs for different systems are built.

3.3 Concept of a SNMP-based real-time data acquisition

Based on the knowledge of the SNMP communication capabilities of the CMLs and respecting the security requirements of CML operators, the concept of our custom CML data acquisition was built on the following findings:

- Necessary CML data can be queried via SNMP.
- SNMP requests can be performed simultaneously for different host systems.

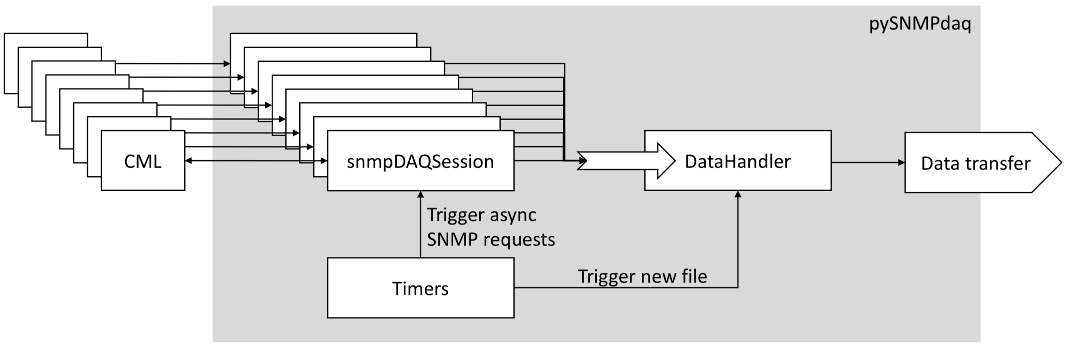

Figure 2. Schematic of the structure of our real-time data acquisition software pySNMPdaq with its three different kind of processes. The inter-process communication is established via `Queues` and `Events` . With the SNMP requests the data flows from the CMLs to the `snmpDAQSessions` and from there to the `DataHandler` , which writes data to files in regular intervals and can transfer the data to a further server.

– SNMP access is only possible from within the private network of the CML operators.

– Access to the private CML operator subnet will not be granted from outside because of security concerns.

After initial testing of a proof of concept and further negotiations with Ericsson Germany the resulting concept, shown in Fig. 1, consisted of

– a data acquisition server located inside the private subnet of the CML operator, running a custom software that performs the SNMP requests

– the custom data acquisition software which is capable of performing simultaneous SNMP requests to a large number of CMLs

– a solution to instantaneously move the data out of the private subnet to a database server running at our institute.

While the concept looks simple, its realization is challenging. The current operational status of the data acquisition is not only a technical but also a superordinate success, since the CML operators had to grant access to their highly secured private network to allow the operation of the custom data acquisition software. Trust is an important prerequisite, for which thorough testing of the software was necessary. We were able to do the necessary tests during development in a non-operational environment with several CMLs at an Ericsson test facility.

4 The implementation: pySNMPdaq

4.1 Overview

Our open-source real-time SNMP-based CML data acquisition software is written in Python and hence was given the name pySNMPdaq.

To assure a constant timing of the simultaneous SNMP requests, pySNMPdaq is divided into three types of subprocesses using the Python standard `multiprocessing` library:

1. `Timer` is the timer `Process` which continuously triggers events at a fixed temporal resolution. The smallest time step is 1 s, with an accuracy of 10 ms. It avoids the typical drift of a simple `while-sleep` loop construct by syncing to the hardware clock of the system it is running on.

2. `snmpDAQSession` is the data acquisition `Process` , one for each CML, which handles the SNMP communication with the CMLs. The OIDs which are used for the SNMP requests are defined in a configuration file and can be different for each `snmpDAQSession` . To assure continuous real-time operation, the SNMP communication is asynchronous and avoids blocking due to failing requests which have not timed out. The SNMP requests are triggered via an `Event` by the timer process. The queried data are put into the `Queue` of the data handler process.

3. `DataHandler` is the `Process` to which all queried data are fed via a `Queue` and which writes the data to file or sends them to an external server via `scp` (secure copy). Writing, closing and transferring files is triggered by the timer via a `Queue` .

Figure 2 shows a schematic of the interplay of the three different process types.

4.2 Requirements

The main requirements, besides software dependencies, to successfully run pySNMPdaq are

– access to a computer with SNMP connectivity to the CMLs

- knowledge of the CML IP addresses
- knowledge of the CML hardware to assure the usage of the correct OIDs for each CML.

To be able to run pySNMPdaq the following Python packages have to be available:

- `pySNMPdaq` (http://github.com)
- `pandas` version $\geq$ 0.14
- `numpy` version $\geq$ 1.8
- `netsnmp` (Python bindings for the net-snmp C-library).

4.3 Application

When the above requirements are met, a list of IP addresses, each with a set of OIDs, has to be compiled and the global configuration file has to be adjusted to define sampling intervals and data handling. We provide examples for both required files in the github repository together with a step-by-step guide on how to generate the listing of IP addresses and OIDs. After the files have been set up correctly, pySNMPdaq can be started and will run as a background process automatically.

When run in an operational CML network, care should be taken to not interfere with its normal operation. Hence a slow increase of the number of CMLs and a reasonable sampling frequency should be considered.

Our test runs at the Ericsson test facility and additional calculation of the caused traffic (see Sect. 6.2) show that a continuous data acquisition of several hundred CMLs at a 1 s interval is theoretically possible and should not impair the network. This has not been tested operationally in a large network, though. A sampling interval of 1 min was agreed upon with Ericsson for the data acquisition of their operational CMLs.

5 Example data

5.1 Data from CMLs for cell phone backhaul network

In summer 2014 we started to run pySNMPdaq operationally on a server in the CML network that Ericsson Germany operates as backhaul for the T-Mobile cell phone net. Slowly increasing the number of simultaneously queried CMLs, we are currently receiving data from 450 Ericsson MINI-LINK TN CMLs every minute with a time delay of only several seconds.

Figure 3 shows a record of one CML over the course of 3 days. The CML has a length of 7.9 km and a frequencies of 22.078 and 23.086 GHz for the two directions. Plotted is the difference between the transmitted and received level (TX-RX level). Quantization is 1 dB for the TX-level and 0.3 dB

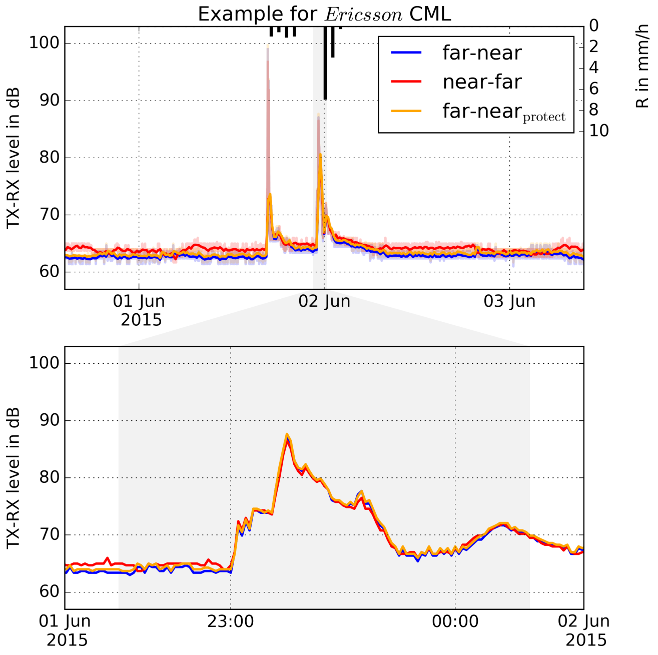

Figure 3. Example data record of an Ericsson MINI-LINK TN CML used in the backhaul of a cell phone network. The CML has a length of 7.9 km and uses frequencies of 22.078 and 23.086 GHz for the two directions. The direction from the far end to the near end uses a second receiver as hot standby 1 + 1 system. Top: the solid lines show the 15 min averages of the TX-RX level (transmitted minus received signal level). The lines with the lighter color show the raw data with 1 min resolution. In addition, hourly rain rates from a DWD rain gauge in the vicinity of the CML are shown (black bars) to indicate the strong sensitivity of the CML path attenuation to rain rate. Bottom: zoomed-in version of the plot above showing only the raw 1 min data.

for the RX-level. The CML is configured as a 1 + 1 protection system using a second transmit-and-receive unit at the near end as a hot standby. This configuration is often used at sites where the outdoor units, which contain the transmitter and receiver, are mounted in places which are difficult to reach. There, an exchange of an outdoor unit is often more expensive than the unit itself.

To give the reader an idea of the impact of rainfall on the attenuation along the CML path, Fig. 3 also shows the hourly records of a DWD (German Meteorological Service) rain gauge, which is located about 4 km from the CML. Please note that the different temporal aggregation and the different spatial coverage of both measurements must lead to differences in the actual observed values and the corresponding time stamps. Nevertheless, a good correlation between path attenuation (TX-RX level) and rain rate is evident. A quantitative analysis, which would require the derivation of rain rates from CML data, is beyond the scope of this article. For a validation of CML-derived rain rates and an explanation of the required processing we hence refer the reader to, e.g., Chwala et al. (2012) or Overeem et al. (2013).

The lower part of Fig. 3 is a zoomed-in version showing one of the strong attenuation events in more detail. One can see that the three channels (far-near, near-far, far-near$_{\mathrm{protect}}$) agree very well. Interestingly there are, however, small differences even between far-near and far-near$_{\mathrm{protect}}$, which operate in the same direction and share the transmitter at the far end. The differences hence must be caused by small variations in the two receivers and the subsequent rounding to the quantized values that are available via SNMP.

5.2 Data from CMLs for ski resort intranet

Besides the operation for the CMLs in the Ericsson cell phone backhaul network, we also installed pySNMPdaq at the Bayerische Zugspitzbahn, the operator of the local alpine ski resort. They connect the summit stations, e.g., on Germany's highest peak, the Zugspitze, with the valley via Ceragon CMLs. Since their network only comprises four CMLs, computational demand was low and pySNMPdaq was run on a low performance netbook placed at the IT center of the Bayerische Zugspitzbahn. Since the operation of this CML network is not as crucial as the one that serves as the cell phone network backhaul, we were allowed to run the data acquisition with a temporal resolution of 1 s.

Figure 4 gives an example of TX-RX level records. In addition, similar to Fig. 3, the hourly rain rates from a nearby DWD rain gauge are shown. Note that the Ceragon CMLs have 1 dB quantization of the received signal level. The transmit level was constant. In the zoomed-in plot the high temporal resolution is revealed. Since data at 1 s intervals are not necessary for rain rate estimation, the quantization could be compensated by averaging. In situations with low fluctuations of the signal level, this however leads to artifacts.

The operation at the Bayerische Zugspitzbahn also proved that pySNMPdaq is capable of acquiring data from different types of CML hardware, since three of the four Ceragon CMLs are IP10 models and one is an IPMax system.

6 Transferability and scalability

6.1 Transferability to other CML networks

As has been mentioned in Sect. 5.2, pySNMPdaq can acquire data for CML networks with heterogeneous hardware as long as the CML hardware supports querying the received, and if necessary the transmitted signal level. To our knowledge this is true for most, if not all, modern CML systems. The only challenge is to derive the correct OIDs for each hardware type.

Given its flexibility regarding CML hardware, pySNMPdaq should be usable in all CML networks worldwide.

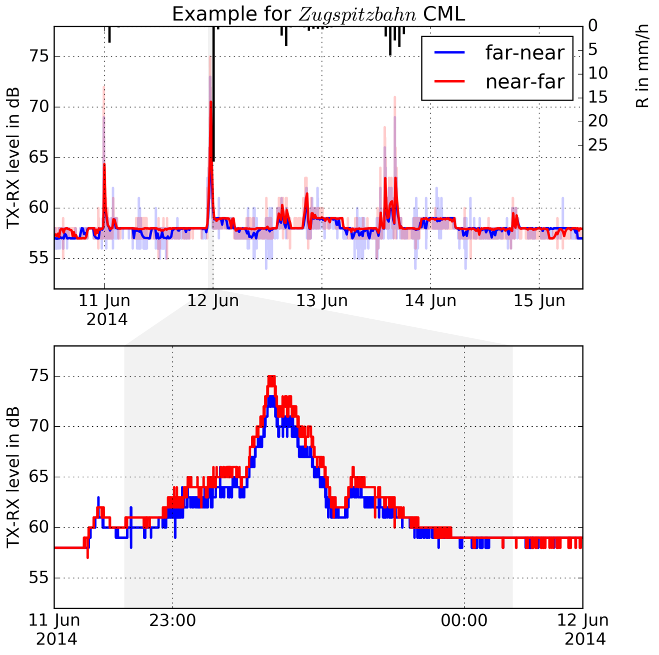

Figure 4. Example data record of a Ceragon IPMax CML used for the intranet connection between summit and valley stations of an alpine ski resort. The CML has a length of 4.7 km and uses frequencies of 22.022 and 23.033 GHz for the two directions. Top: the solid lines show the 15 min averages of the TX-RX level (transmitted minus received signal level). The lines with the lighter color show the raw data with 1 s resolution. In addition, hourly rain rates from a DWD rain gauge in the vicinity of the CML are shown (black bars) to indicate the strong sensitivity of the CML path attenuation to rain rate. Bottom: zoomed-in version of the plot above showing only the raw 1 s data.

6.2 Consideration regarding data traffic

Since it is crucial that the data acquisition software does not impair the stability of the SNMP communication within an operational CML network, the additional data traffic must not overload the network. To estimate the data traffic limits it is important to know the SNMP bandwidth of the CMLs and the topology of the CML network.

To our knowledge the CMLs have a SNMP service channel with a bandwidth between 64 and 192 kbps s^{-1}. A typical SNMP request of four values (TX- and RX-levels for the near end and the far end) causes a data traffic of approximately 500 bytes, 250 bytes for the request, 250 bytes for the response, mostly caused by the package overhead. That is, even the slowest SNMP service channel with 64 kbps, equal to 8 kilobytes per second, could theoretically handle 16 of those requests every second.

Hence, as long as the number of SNMP requests that are routed via one CML is small, there should not be any problems with the additional data traffic. This will however depend on the topology of the network. The CML network of Ericsson in Germany for example is structured in small hubs of CMLs which are connected via DSL to the IT center. That is, there is no big accumulation of SNMP requests to one or very few CMLs. For other networks, that may differ and it is up to the future user of pySNMPdaq to evaluate the network structure and use a safe number of parallel requests and a safe sampling rate.

Given a certain upper boundary for the number of simultaneous SNMP requests, the number of CMLs that are included in the data acquisition could be raised by staggering the requests into batches. For example, 1000 simultaneous requests could be sent out every 5 s, each time for 1000 different CMLs, but repeated in the same order every minute. This way, not all CMLs will be sampled simultaneously, but the sampling rate for each individual CML would still be constant.

The current implementation of pySNMPdaq does not yet support staggered simultaneous requests. We will add this feature in the coming month, though, and it will be made available with the updated version in the github repository.

6.3 Scaling of computational demand

The maximum number of simultaneous SNMP requests is not only limited by the bandwidth constraints of the CML network. Each parallel `Process` of the operating system and each interprocess communication data structure like a `Queue` or an `Event` requires additional memory.

Figure 5 shows the memory usage of the current implementation of pySNMPdaq for different numbers of parallel processes, i.e., simultaneous SNMP requests. We tested on a machine with only 8 GB of RAM, but the fitted second-order polynomial yields a memory usage of approximately 7.7 GB for 1000 parallel processes and of approximately 25.8 GB for 2000 parallel processes. The memory usage could probably be optimized by exchanging parts of the current interprocess communication with a low-level solution, e.g., by using flags in shared memory. We do, however, not see the current need to optimize the memory usage, because the CML network operators will want to keep a safety margin regarding data traffic. Hence, they will limit the number of simultaneous SNMP requests anyway, in our case to around 1000.

To overcome both the computational demand and the data traffic limitations, a solution for large and countrywide CML networks could also be to use several data acquisition servers which are spatially distributed across the network.

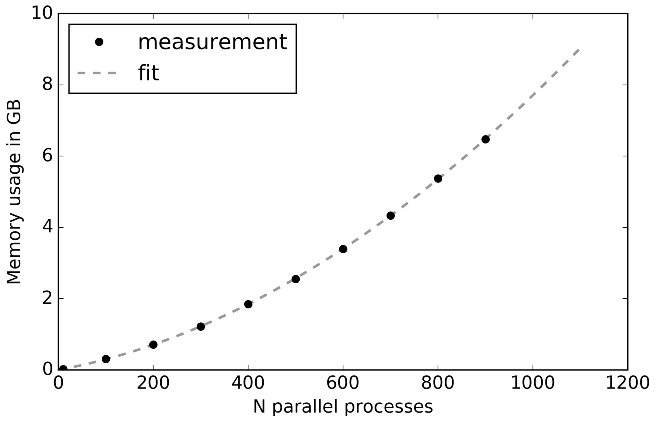

Figure 5. Memory usage of pySNMPdaq for different numbers of parallel processes, i.e., simultaneous SNMP requests. The `Processes` alone contribute linearly, while their interprocess communication via `Events` contributes quadratically. A least square fit of the memory usage $\mathrm{mem_{GB}}$ in gigabytes yields $\mathrm{mem_{GB}} = 2.5 \times 10^{-3}\ \mathrm{N_{processes}} + 5.2 \times 10^{-6}\ \mathrm{N^2_{processes}}$.

7 Conclusions

We have introduced our newly developed operational real-time data acquisition system for CML data, which is available as open-source software at http://github.com/cchwala/pySNMPdaq. This system is the basis for continuous countrywide access to CML data for line integrated precipitation quantification.

While others have already shown that it is possible to run a specifically designed data acquisition software for a small number of CMLs within an operational network (Fencl et al., 2015), our application is the first where hundreds of operational CMLs are queried simultaneously and with a constant sampling rate. Our software is also capable of acquiring data from different CML hardware models and manufacturers and, hence, is applicable in any CML network.

We have shown that it is possible to provide CML data with a constant and high temporal resolution (1 min or better) and in near-real time (only several seconds delay).

In particular the almost instantaneous availability is a crucial prerequisite for future operational applications of CML data for quantitative precipitation estimation (QPE). Retrospective and theoretical studies have already shown the potential of CMLs to improve QPE from weather radar (Trömel et al., 2014) and to considerably increase the accuracy of urban drainage modeling (Fencl et al., 2013). But the CML data must be available at the time the rain is falling to be able to predict and protect. This is now possible.

For Germany, the next step is a further extension of the data acquisition system to provide a countrywide coverage using several thousand CMLs. Furthermore, negotiations have been initialized with cell phone providers in Palestine and Burkina Faso (Gosset et al., 2015) to improve the data acquisition of CMLs for rain rate estimation in these data-scarce regions.

Appendix A: Example OIDs for received signal level

This Appendix describes the OIDs for requesting the received signal level for the CML hardware that was used to record the data shown in Sect. 5.

Ericsson MINI-LINK Traffic Node systems support the operation of several CMLs as subsystems of one main control unit. The different CML subsystems are defined by their slot number. On top of that it has to be defined whether the near end or the far end should be queried. Building on the base OIDs for requesting the RX-level, 1.3.6.1.4.1.193.81.3.4.3.1.3.1.10, the additional information is encoded in the so-called interface descriptor. As already observed by Wang et al. (2012), the interface descriptor is a cryptic number, e.g., 214 669 7473 for slot number 2 at the near end. The logic behind the cryptic number is revealed when different interface descriptors are given in hexadecimal representation. Table A1 lists OIDs and interface descriptors for several Ericsson MINI-LINK Traffic Node subsystems. Going from slot 2 at the near end to slot 3 at the near end the interface descriptor changes from `0x7ff40101` to `0x7ff40181`. This is equal to a switch of bit number 8 from 0 to 1. Similarly the difference between near end and far end is a switch of bit number 19 from 1 to 0.

Other CML hardware uses different OIDs and different methods to distinguish the subsystems. For example a Ceragon IPMax uses gnOduStatusXReceiveLevel as base OID to query the current received signal level. It supports two subsystems, in drawer1 and drawer2, for which the base OID has to be altered. Table A1 gives these OIDs together with that of the older Ceragon IP10 system which does not support subsystems.

To identify the required OIDs for other CML systems, one has to consult the management information base (MIB) and the corresponding documentation which define all OIDs for a certain system.

Table A1. Variations of the OIDs for requesting the current received signal level for different CML systems and, if applicable, for their subsystems. The Ericsson MINI-LINK Traffic Node (TN) uses an interface descriptor to identify the subsystems and is able to also access data from the far end of the CML. The Ceragon IPMax systems support only two subsystems in separate drawers, while the Ceragon IP10 does not support subsystems.

Hardware	OID (as text)	OID and interface descriptor (as number)	Interface descriptor (as hex)
Ericsson MINI-LINK TN	xfRFCurrentInputPower (slot 2 near)	1.3.6.1.4.1.193.81.3.4.3.1.3.1.10.2146697473	0x7ff40101
	xfRFCurrentInputPower (slot 2 far)	1.3.6.1.4.1.193.81.3.4.3.1.3.1.10.2129920257	0x7ef40101
	xfRFCurrentInputPower (slot 3 near)	1.3.6.1.4.1.193.81.3.4.3.1.3.1.10.2146697601	0x7ff40181
	xfRFCurrentInputPower (slot 3 far)	1.3.6.1.4.1.193.81.3.4.3.1.3.1.10.2129920385	0x7ef40181
	xfRFCurrentInputPower (slot 4 near)	1.3.6.1.4.1.193.81.3.4.3.1.3.1.10.2146697729	0x7ff40201
	xfRFCurrentInputPower (slot 4 far)	1.3.6.1.4.1.193.81.3.4.3.1.3.1.10.2129920513	0x7ef40201
Ceragon IPMax	gnOduStatusXReceiveLevel.drawer1	1.3.6.1.4.1.2281.3.1.5.1.5.3	–
	gnOduStatusXReceiveLevel.drawer2	1.3.6.1.4.1.2281.3.1.5.1.5.4	–
Ceragon IP10	genEquipRfuStatusRxLevel	1.3.6.1.4.1.2281.10.5.1.1.2.1	–

Acknowledgements. We want to gratefully thank our cooperation partners at Ericsson Germany: Jürgen Fritz for the support with our first data acquisition based on the data loggers and for introducing us to the right people to make the move to SNMP possible; Jürgen Alschbach for the technical support while figuring out the details of the CML SNMP connectivity in the test lab; Ronald Riedel for supporting our ideas and thus for making it possible to move from the lab to the operational network; and Reinhard Gerigk and Declan Forde for the initial and continuous support with the actual operational system.

Furthermore we want to thank Christian Scharpf from Bayerische Zugspitzbahn AG for making it possible to record CML data in the alpine terrain in our backyard.

This work was supported by the German Research Foundation (DFG) through the project "Integrating Microwave Link Data For Analysis of Precipitation in Complex Terrain: Theoretical Aspects and Hydrometeorological Applications" (IMAP). Further support was provided by the Stiftung Energieforschung Baden-Württemberg.

The article processing charges for this open-access
publication were covered by a Research
Centre of the Helmholtz Association.

Edited by: S. Buehler

References

Chwala, C., Gmeiner, A., Qiu, W., Hipp, S., Nienaber, D., Siart, U., Eibert, T., Pohl, M., Seltmann, J., Fritz, J., and Kunstmann, H.: Precipitation observation using microwave backhaul links in the alpine and pre-alpine region of Southern Germany, Hydrol. Earth Syst. Sci., 16, 2647–2661, doi:10.5194/hess-16-2647-2012, 2012.

David, N., Alpert, P., and Messer, H.: Technical Note: Novel method for water vapour monitoring using wireless communication networks measurements, Atmos. Chem. Phys., 9, 2413–2418, doi:10.5194/acp-9-2413-2009, 2009.

Doumounia, A., Gosset, M., Cazenave, F., Kacou, M., and Zougmore, F.: Rainfall monitoring based on microwave links from cellular telecommunication networks: First results from a West African test bed, Geophys. Res. Lett., 41, 6016–6022, 2014.

Fencl, M., Rieckermann, J., Schleiss, M., Stránský, D., and Bareš, V.: Assessing the potential of using telecommunication microwave links in urban drainage modelling, Water Sci. Technol., 68, 1810, doi:10.2166/wst.2013.429, 2013.

Fencl, M., Rieckermann, J., Sýkora, P., Stránský, D., and Bareš, V.: Commercial microwave links instead of rain gauges: fiction or reality?, Water Sci. Technol., 71, 31–37, 2015.

Gosset, M., Kunstmann, H., Zougmore, F., Cazenave, F., Leijnse, H., Uijlenhoet, R., Chwala, C., Keis, F., Doumounia, A., Boubakar, B., Kacou, M., Alpert, P., Messer, H., Rieckermann, J., and Hoedjes, J.: Improving Rainfall Measurement in gauge poor regions thanks to mobile telecommunication networks, Bull. Am. Meteorol. Soc., in press, doi:10.1175/BAMS-D-15-00164.1, 2015.

Leijnse, H., Uijlenhoet, R., and Stricker, J. N. M.: Rainfall measurement using radio links from cellular communication networks, Water Resour. Res., 43, W03201, doi:200710.1029/2006WR005631, 2007.

Messer, H., Zinevich, A., and Alpert, P.: Environmental Monitoring by Wireless Communication Networks, Science, 312, 713 pp., doi:10.1126/science.1120034, 2006.

Overeem, A., Leijnse, H., and Uijlenhoet, R.: Country-wide rainfall maps from cellular communication networks, Proc. Natl. Acad. Sci., 110, 2741–2745, 2013.

Trömel, S., Ziegert, M., Ryzhkov, A. V., Chwala, C., and Simmer, C.: Using Microwave Backhaul Links to Optimize the Performance of Algorithms for Rainfall Estimation and Attenuation Correction, J. Atmos. Oc. Technol., 31, 1748–1760, 2014.

Wang, Z., Schleiss, M., Jaffrain, J., Berne, A., and Rieckermann, J.: Using Markov switching models to infer dry and rainy periods from telecommunication microwave link signals, Atmos. Meas. Tech., 5, 1847–1859, doi:10.5194/amt-5-1847-2012, 2012.

Zinevich, A., Alpert, P., and Messer, H.: Estimation of rainfall fields using commercial microwave communication networks of variable density, Adv. Water Resour., 31, 1470–1480, 2008.