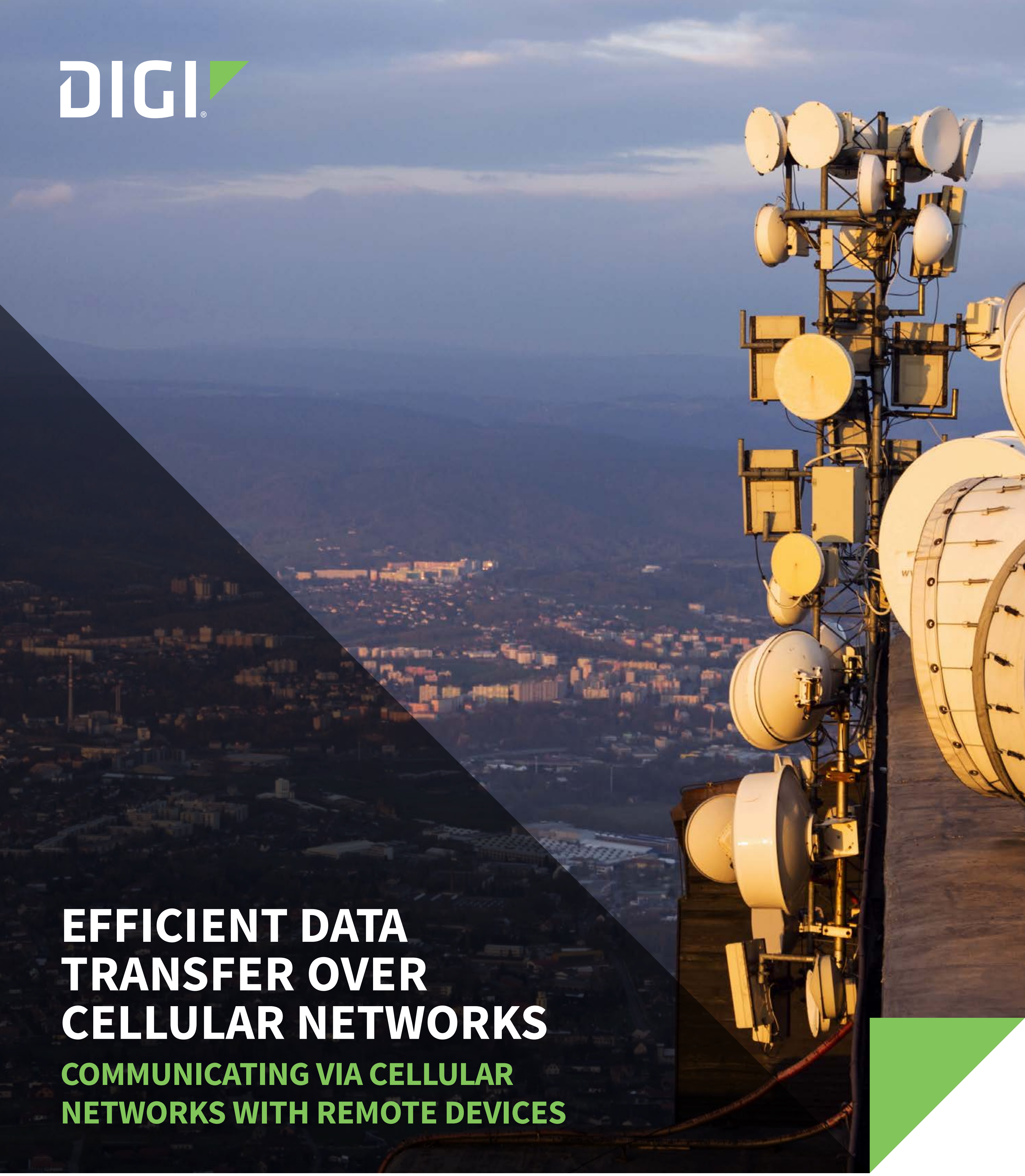


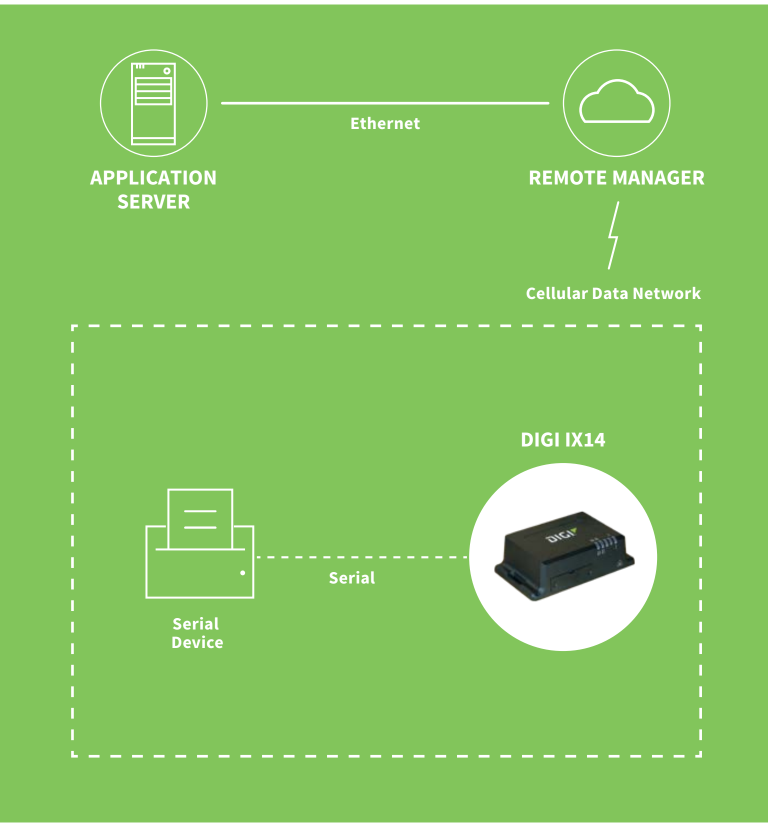
EFFICIENT DATA TRANSFER OVER CELLULAR NETWORKS

COMMUNICATING VIA CELLULAR NETWORKS WITH REMOTE DEVICES

1. Cellular Data Transmissions

GENERAL OVERVIEW

Cellular data transmission is an increasingly attractive mechanism for communication with remote, non-permanent or mobile devices. Cellular networks enable greater efficiency and reliability for enterprises to collect and distribute data virtually anywhere. The fact that cellular data is metered means that the frequency of transmission and amount of data sent in each exchange can have significant cost and performance impact. However, the cost of cellular data plans is on the decline and network technology is improving.



In order to understand this impact, let us start with a fairly typical example, where there is a device in the field and an application on a server at a central site location that collects information from that device.

In general, the purpose of communication with the device will be for one of two reasons:

- **Monitoring data** – Information such as the level or temperature of a storage tank, the velocity and pressure of a pipeline, the condition of a controller or the status and uptime of a remote device
- **Transaction data** – Discrete event data, such as cash or credit transactions, PBX call records or mission-critical/safety-related alarms

Monitoring data is often “polled.” The application sends out periodic queries and gets responses for those queries. The application can usually retry if it does not get an answer, and determine that a problem exists if it does not get a response after a certain amount of retries.

Discrete event data is usually “unsolicited.” The application does not expect to get information on any regular basis, and therefore the failure to hear from the device is the normal case (though some sort of “all is well” message may be sent at a longer interval).

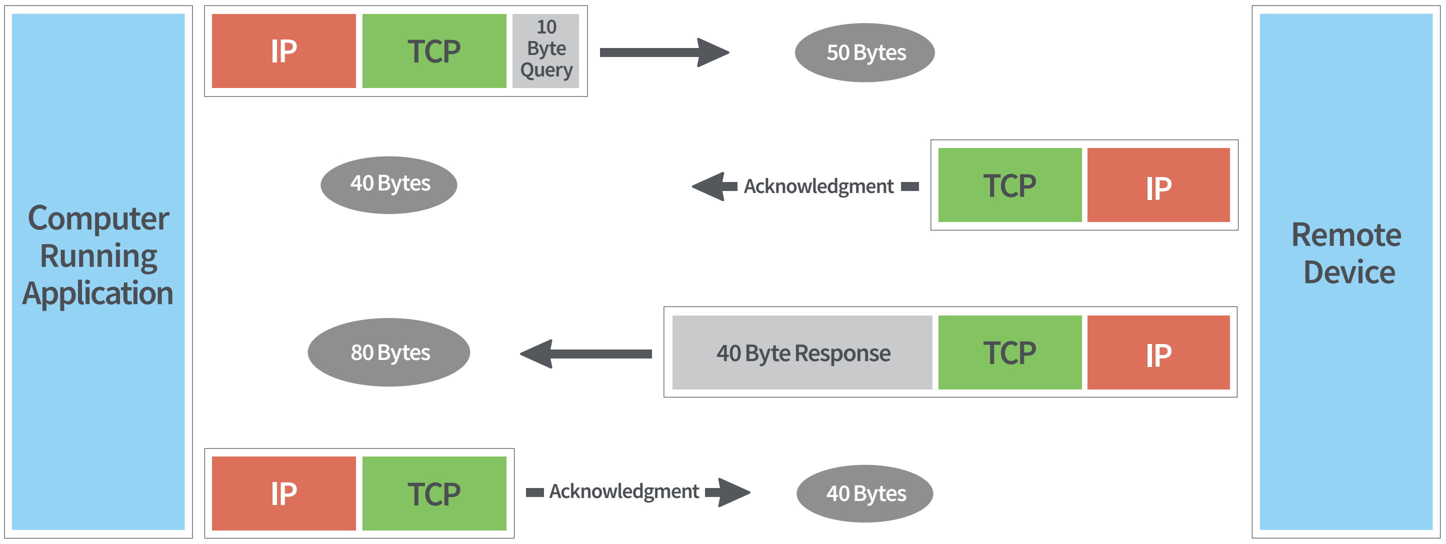
Most applications will likely involve one or both of these methods.

COMPUTING PROBABLE DATA TRANSFER REQUIREMENTS

In cellular data networks, data is transmitted in TCP or UDP packets comprised of a data payload of one or more bytes and a set of additional bytes called a header that is attached to the data payload. Because of these header bytes, smaller data transmissions incur a higher percentage of overhead.

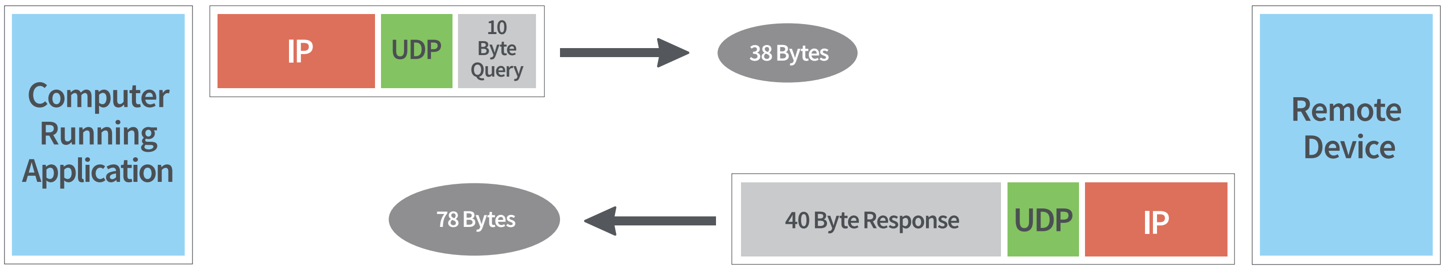
Typical TCP Query and Response Transaction

(Note the two additional acknowledgement packets required.)



In general, a TCP/IP header has 40 bytes, and each TCP/IP packet sent will generate a reply acknowledgment (ACK) packet of 40 bytes with no data inside of it. If a packet is not acknowledged, the TCP/IP packet containing the data will be resent. The rate of retransmission of packets depends on the reliability of the underlying network and the configuration of the TCP stack. Additionally, the application itself may attempt to resend the data, so even if the TCP/IP stack discards the packet due to a network timeout condition, the application itself may send the data again, causing a new packet to attempt to propagate across the network.

Typical UDP Query and Response Transaction



In general a UDP/IP packet will have 28 bytes of header data; however, UDP packets are only sent one time, and there is no ACK. UDP therefore has at least a 50 percent advantage in overhead on a highly reliable network. If the application or data device will resend or re-query for data, UDP can offer significant savings in terms of network efficiency.

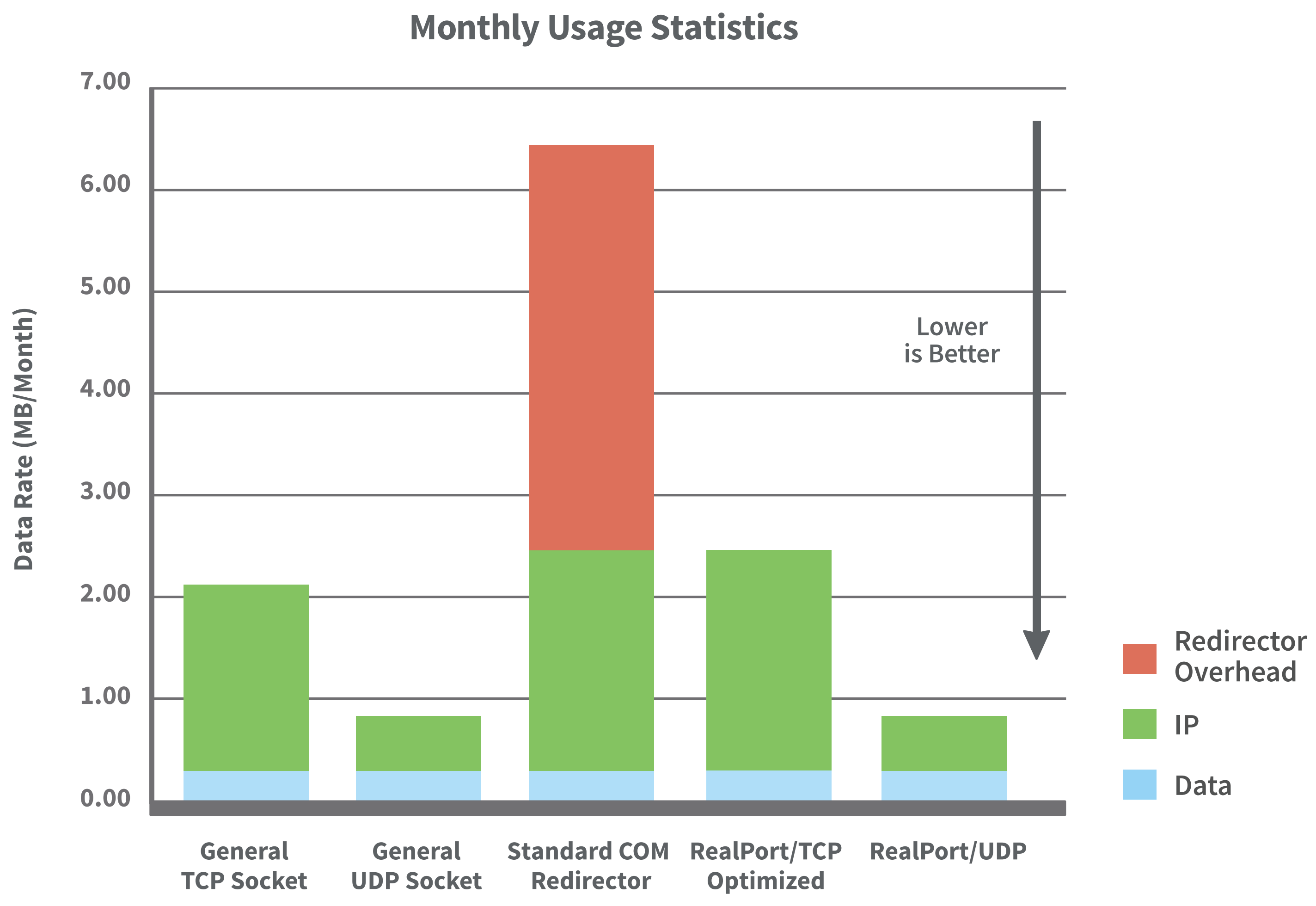
A PRACTICAL EXAMPLE

In our initial scenario, assume that the device in question is a PLC using Modbus protocol that gets polled from a central location once every five minutes. Depending on the particulars of the implementation, the connection could be TCP or UDP and may require use of COM port redirection software such as Digi's RealPort® driver. In this scenario there are widely differing levels of overhead, depending on the precise configuration of the implementation. A spreadsheet for performing similar calculations is available from Digi International.

In this scenario, the range of usage for the device might be:

- **<1 MB/month in the UDP socket or RealPort/UDP case**
- **2-3 MB/month for TCP socket or RealPort/TCP optimized for cellular**
- **6.5+ MB for a standard COM port redirector**

The design of cellular network architecture requires implementers to carefully consider their bandwidth needs and application/connectivity requirements in order to achieve maximum benefit and meet cost targets. Part II of this document analyzes the different components involved in cellular communication to help the implementing engineer determine the best approach for the specific application.



II. Technical Considerations in Evaluating Cellular Communications

GENERAL OVERVIEW

Three broad technical areas affect planning and implemention of a cellular data application:

- **Network performance constraints**
- **Network overhead**
- **Application performance**

While these technical areas are reasonably well defined in the cellular network specifications, most of the available literature is focused at the theoretical or network architecture level, or alternately assumes the deployment of consumer-facing devices and uses such as browsing from a laptop, tablet or mobile phone.

To understand the implications of deploying remote devices we have to look at the information in detail. Let us evaluate each of these areas in terms of our basic scenario of a remote field device and a central-site application.

NETWORK PERFORMANCE CONSTRAINTS

Network Coverage

Widely deployed devices may have coverage requirements that are not satisfied by any one carrier. For example, some carriers may not support the ability to provide static IP addresses.

Carriers are ultimately subscriber-focused. They focus their efforts on updating their infrastructure first in areas where the number of subscribers is greatest. Remote devices that need cellular coverage are not always deployed in these zones, so it is important to work closely with the cellular service providers and cellular hardware vendors to coordinate rollout of cellular devices based on available and planned infrastructure.

Network Reliability

No single technology solution can offer 100 percent reliability. Cellular technologies are resilient against most “backhoe” type issues such as physical disruption of landlines. In many cases, landlines provided by different carriers can travel in a common conduit or through a common collocation facility. Physical diversity of connection is a significant strength of cellular data networks. Cellular networks are also substantially more resistant to rain-fade and cloud cover than most satellite networks. Cellular data networks are also usually segregated from voice/circuit based connections, so that even when “all circuits are busy” for voice, cellular data can still get through.

Nonetheless, cellular connections can still experience momentary cellular network dropouts and data loss, so while cellular data is “highly available,” delivery is not 100 percent guaranteed.

Network Imposed Latency

Performance of data transfer across cellular networks can vary significantly based on carrier coverage, type of network, and activity on the network at the time of transmission. Voice calls have lower latency than data calls because voice calls can lose significant quantities of data before comprehension is degraded. Data calls are routed through a separate network to improve data fidelity, at the expense of significant latency imposed by the cellular carrier’s network architecture.

Available Network Bandwidth

As stated earlier, cellular carriers are subscriber-focused. In addition to deploying the latest technology first in areas with the greatest number of subscribers, the network is also prioritized for customer-facing devices. For example, an HTTP request to a website is much smaller than the download of the requested file. Downloading email is similar. Most cellular networks, particularly the first releases of 3G technology, were designed to maximize download (meaning toward the mobile device) versus upload performance.

Remote device connectivity would typically have better performance if this was reversed. A central site sending a query to a remote device will almost always send less data (downstream across the cellular connection) than it expects to receive in reply. Likewise, in cases where the data is sent from the field locations unsolicited to the central site, the primary data flow is upstream.

Therefore, when connecting remote devices to the network, it is important to remember that marketing literature will refer more prominently to the download speed.

GENERATION	BANDWIDTH (DL)	LATENCY
2G	< 100 kbps	250 ms - 1 s
3G	100 kpbs - 1 Mbps	100 ms - 250 ms
4G CAT1	1 Mbps - 5 Mbps	~50 ms - 100 ms
4G LTE-M	10* - 100 - 250 kbps	~50 ms - 2 s - 4 s*
4G NB-IoT	~25 kbps	~100 ms - 4 s - 10 s*
5G	kbps - Gbps	1 ms - seconds*

* Coverage Enhancement (CE) mode increases range, but lowers bandwidth and increases latency

NETWORK OVERHEAD

The architecture of the network as described above means that there is overhead to be measured based on how the application operates across the transport mechanism, and how often communications are re-tried when the end device is non-responsive. Additionally, various security measures may also increase network overhead.

Data Transmission Overhead

There are two primary methods of communication within an IP-based network such as a cellular data network: TCP and UDP. Each has its benefits and limitations. Selecting the correct approach for a given application is the single greatest contributor to implementing a cost-effective and satisfactory solution.

It is often stated that TCP is reliable — it guarantees delivery of data — while UDP is unreliable. While this is technically (mostly) true, it is more accurate to think of the difference as follows: TCP provides a mechanism within the network layer to identify and attempt to retransmit dropped packets, while UDP leaves that up to the application layer.

Many applications, especially in the device monitoring space, already have mechanisms in place to validate data integrity and request retransmission of missing information. Reliable data communications are quite commonly implemented over UDP using these applications. Letting the application handle the data integrity/retransmission often results in considerable cost savings and can sometimes even reduce latency end-to-end.

Other applications deliberately over-sample. In other words, they request or send information more often than is actually necessary, so that if any one communication is lost, the purpose for which the data is being collected is still satisfied.

However in other cases, TCP is a requirement because there is no satisfactory mechanism in the device or in the application to store data, or every transmission must definitively succeed or fail (e.g., cash transaction-related data flows for selling lottery tickets).

Network Disconnections and Duplicated Requests

A dropped packet will generate multiple TCP retransmissions by default before the data is discarded. If the polling rate is less than the TCP timeout, the application might send multiple requests for the same information across the network, and occasionally receive double responses from the device. In general, the TCP keep-alive should not be set for a value greater than the amount of time the application will continue to listen for a valid response.

Each time a TCP connection is re-established or torn down, additional traffic is generated. Keep-alive packets that ensure connections remain active also generate traffic. For example, four keep-alive packet transmissions take as much data as one close and re-open connection. (Specifically, it takes one packet to close the connection and three to open it.) If keep-alive packets are sent every 4.5 minutes, it typically makes sense to keep connections open that will poll every 15 minutes. It may be more efficient to close and re-open connections that poll less frequently.

Security Concerns

Cellular data is transmitted in an encrypted form while traveling over the air, but once the data reaches the wireline network of the cellular carrier it is normally transmitted in the clear. Some applications require that data be encrypted while traveling on the network (sometimes just on public networks, sometimes end-to-end between sending and receiving equipment).

There are several methods of providing this encryption:

- TCP Packets can be encrypted through an SSH tunnel.
- All traffic can be encrypted through a VPN tunnel.
- Specific protocol traffic can be encrypted through use of an SSL/TLS method.
- HTTPS (secure web) traffic uses TLS/SSL to encrypt traffic from a web browser/server.
- Digi's RealPort driver when running in TCP mode provides a method for encrypting traffic using TLS/SSL without the use of a separate VPN connection.

All three of these methods increase the overhead of the data being sent:

- The underlying network will not transmit the data in an exact packet-for-packet manner.
- Every time the connection is initiated and/or torn down, additional data will be retransmitted to establish and end the security context.
- The encrypted communications will normally pad the data being sent to a certain extent, once the communication channel is set up.

Using IPsec ESP Tunnel Mode, and the 10-byte query, 40-byte response packet from the TCP and UDP diagrams in the first section, each packet would increase in size by 64-84 bytes, effectively doubling the amount of data sent over the connection. In practice, the worst case seems to be about 110-115 percent, so for budgetary purposes it may be sufficient to double the unencrypted traffic estimate and use that as a baseline to estimate the amount of encrypted traffic.

APPLICATION EXPECTATIONS FOR DATA TIMING

In theory, any TCP/IP (or UDP/IP) based protocol will work fine over an IP-based Wide Area Network (WAN). However, implementers often unconsciously build in Local Area Network (LAN) timing assumptions that prevent their products from running successfully over a WAN. In general, satellite and cellular networks require software to be patient. Prematurely timing out and retrying when the network is busy makes matters worse and can actually prevent successful communications.

Unfortunately, most product developers only test on Ethernet/LAN; it is very possible the first users attempting to use a WAN will need to locate and point out the problems for the developers.

Connection Delay

Most applications use the operating system defaults; on Microsoft Windows this is generally five seconds. Applications may not provide a mechanism to modify this default, so users may not have any option to change this default behavior.

In the best-case scenario, not waiting long enough to open a socket just makes reconnection difficult at times. As long as the application waits at least 30 seconds before it retries the connection, it will eventually recover. However, the worst-case occurs if the application not only times-out too fast, but retries too fast. The TCP peers alternate between thinking they are connected but having to "reset" the connection due to timeouts and thinking they need to retry the connection; this could continue for as long as the WAN is congested.

Response Delay
Many applications by default assume Ethernet/LAN responses occur in 250 milliseconds or less. Fortunately, most applications allow users to change this value. Unfortunately, some applications limit the maximum response delay to five or 10 seconds. A WAN-aware application should allow this setting to be at least 30 seconds – preferably at least 60 seconds.

Besides the obvious performance problem of too many timeouts repeatedly putting the remote device "offline," a more risky problem is how the application will handle unexpected responses (technically, "no-longer expected" responses). A simple example is an application that sends a request, then times-out twice and tries twice. How will the application react when it receives three responses at the same time? Remember, the first two requests were not lost; they still reached the remote device. Their responses were just delayed longer than expected.

Cost to Communicate
Few applications are written to optimize network traffic; after all, it is usually the end devices themselves and not the "Ethernet" that is the limiting factor. But put such an application across a WAN and you may discover that 99 percent of the data you are paying for is either protocol overhead or data updates without any change in data. Here are some examples of monthly data usages based on 200 byte transactions.

200 bytes per second	**= 518 MiB/month**
200 bytes per 5 seconds	**= 86 MiB/month**
200 bytes per minute	**= 9 MiB/month**
200 bytes per hour	**= 0.14 MiB/month**

Note: Carriers typically round up usage to the nearest kilobyte every hour.

Idle TCP Sockets
This is related to Cost to Communicate. The obvious solution to reduce cost is to slow down data polls. However, if idle time between transmissions on TCP sockets becomes too long, the ability of an application or the network stack to keep track of valid sockets can become compromised because it may run into a transient network outage, or the device on the other end may have become non-responsive in a way that prevents replying to a packet. It will send a packet, wait, and see no ACK or other indication the socket is closed; then it will follow the normal TCP rules of back-off and retry.

This issue likely varies based on WAN technology, but a good rule of thumb at present is that it is necessary to either send data or a TCP keep-alive every four to five minutes to keep the TCP socket healthy.

UDP Reliability
An application using one or two UDP packets per transaction will be unlikely to experience WAN-related issues. The big problem comes with applications that require tens of thousands of sequential UDP packets to complete a single transaction, such as TFTP for file transfer. The longer response lags and higher probability of UDP packet loss may prevent the application from ever completing the transaction.

To minimize cost, applications may need to be rewritten to implement Report-By-Exception or Report-Only-on-Change-of-State.

Review of Initial Scenario
In cases where the data source is queried for current status data, it is usually true that the data-requesting application can resend a query if it does not receive an answer or if the answer is not valid (due to factors such as corruption of data in transmission or loss of link). As long as the number of unanswered or incorrectly answered queries is small, resending the occasional query is significantly more efficient than validating all the data sent and received.

In cases where unsolicited data is received from the field, the central site cannot know about data it has not received, though it may be able to identify and reject invalid messages. In this sort of scenario, the field device has to verify that the information it sent was received by the central site before it can clear the message from its memory. Mechanisms for retransmission of the message and/or for sending the message to a secondary location if a valid message receipt is not returned are necessary features of such a solution.

The advantage of field-initiated communications is that central site computing resources and applications may only have to scale as a function of the number of exceptions rather than the number of field devices.

It may also be important to factor in the bandwidth requirements for firmware upgrades to remote units. Firmware upgrades will either use TCP (such as through a web page) or UDP (such as through TFTP). In either case, assuming the network is stable, the bandwidth required to upgrade firmware will usually be no more than 150 percent of the size of the actual firmware image, since the firmware will be sent in relatively large packets. Actual performance in some cases may be much better than this.

The appendices to this document include a more detailed description of the network stack involved in cellular communications and a chart of typical usage levels for small-byte-count communications at different intervals.

SUMMARY

Cellular data connectivity provides a powerful tool for automation of data collection and redundant communications. Understanding the particulars of cellular technology can ensure you are taking the best approach to optimize bandwidth and cost.

IP ADDRESSING TYPES

There are two basic approaches to cellular communications: Mobile Originated and Mobile Terminated. These can sometimes be aggregated (Mobile to Mobile) and are available from different carriers under different plans.

Mobile Originated

Mobile Originated addresses require communications to be initiated by the field device. This is the type used for most customer-facing mobile equipment such as cell phones and laptop cards. The IP address is often a private address behind a NAT service.

Mobile Terminated

Mobile Terminated devices receive an IP address (either fixed dynamic or permanent) and can receive communications initiated from a central server. In the cases where the address is fixed dynamic, some sort of DDNS service will provide consistent access via a DNS name.

Because Mobile Originated connections register only when they are active on the network, if a Mobile Originated device goes offline, the network will eventually stop trying to forward data to the device, limiting the amount of data charge for attempts to reach the device while it is offline.

Since the network must always assume that the Mobile Terminated connection may be reachable, it does not block communications attempt, even when the device is offline, the effect of which is that polling attempts to reach an offline device will result in chargeable data. In some cases, this billable data during unreachable periods may be significantly larger than the data that would be sent during times when the remote device is available. The amount of potentially billable data sent to a Mobile Terminated device must be controlled by the sending software. Typically this is done through back-off mechanisms (if no response is received, wait a longer interval and try again).

An additional limitation of Mobile Terminated connections is that any device that is not administratively blocked (e.g., by a firewall) from attempting communication with the cellular device can also cause chargeable data events. This is particularly of concern with regard to viruses and denial of service type attacks. Control of the network on which the devices are located is thus of critical importance.

Need help? Find Professional Services at www.digi.com.

Generic TCP socket: 40-byte header, ACK included in data packet

Packet Size Bytes	Frequency 1 min	5 min	15 min	30 min	1 hr
1	5.27	1.82	1.12	0.64	0.32
10	6.05	1.98	1.17	0.66	0.33
20	6.91	2.15	1.23	0.69	0.35
30	7.78	2.32	1.29	0.72	0.36
50	9.50	2.67	1.40	0.78	0.39
100	13.82	3.53	1.69	0.92	0.46

Generic TCP socket: 40-byte header, ACKs in separate packets from data

Packet Size Bytes	Frequency 1 min	5 min	15 min	30 min	1 hr
1	7.00	2.17	1.23	0.69	0.35
10	7.78	2.32	1.29	0.72	0.36
20	8.64	2.50	1.34	0.75	0.37
30	9.50	2.67	1.40	0.78	0.39
50	11.23	3.01	1.52	0.84	0.42
100	15.55	3.88	1.80	0.98	0.49

Generic UDP socket: 28-byte header, no ACK packets

Packet Size Bytes	Frequency 1 min	5 min	15 min	30 min	1 hr
1	2.51	0.50	0.17	0.08	0.04
10	3.28	0.66	0.22	0.11	0.05
20	4.15	0.83	0.28	0.14	0.07
30	5.01	1.00	0.33	0.17	0.08
50	6.74	1.35	0.45	0.22	0.11
100	11.06	2.21	0.74	0.37	0.18

Generic TCP-based COM port redirector, not optimized for cellular data

Packet Size Bytes	Frequency 1 min	5 min	15 min	30 min	1 hr
1	27.13	6.19	2.58	1.37	0.68
10	27.91	6.35	2.63	1.39	0.70
20	28.86	6.54	2.69	1.42	0.71
30	29.72	6.71	2.75	1.45	0.73
50	31.45	7.06	2.86	1.51	0.75
100	35.77	7.92	3.15	1.65	0.83

Digi International RealPort driver (TCP version) with cellular data optimization

Packet Size Bytes	Frequency 1 min	5 min	15 min	30 min	1 hr
1	7.08	2.18	1.24	0.70	0.35
10	7.86	2.34	1.29	0.72	0.36
20	8.81	2.53	1.36	0.75	0.38
30	9.68	2.70	1.41	0.78	0.39
50	11.40	3.05	1.53	0.84	0.42
100	15.72	3.91	1.82	0.98	0.49

Digi International RealPort driver (UDP version)

Packet Size Bytes	Frequency 1 min	5 min	15 min	30 min	1 hr
1	2.51	0.50	0.17	0.08	0.04
10	3.28	0.66	0.22	0.11	0.05
20	4.15	0.83	0.28	0.14	0.07
30	5.01	1.00	0.33	0.17	0.08
50	6.74	1.35	0.45	0.22	0.11
100	11.06	2.21	0.74	0.37	0.18

Assumes sent and received data amounts are the same. To calculate monthly usage for data where the sent and received data are different, take half of the resulting number for each size (sent and received) and add the two halves together.

Assumes no network outages or retransmission of dropped packets.

Assumes unencrypted data.

Assumes for polling intervals *less* than 15 minutes, that keep-alives are set for 4.5 minutes and that the connection is kept open.

Assumes, for polling intervals *greater* than 15 minutes, that keep-alives are not used and that the connection is closed between polls.

To perform more detailed calculations, contact Digi Sales for our spreadsheet calculator.

Contact a Digi expert and get started today

PH: 877-912-3444
www.digi.com

Digi International Worldwide HQ
9350 Excelsior Blvd. Suite 700
Hopkins, MN 55343

/digi.international

@DigiDotCom

/digi-international