Scalability and Development of Space Exploration Networks

Vincenzo Liberatore
Division of Computer Science Case Western Reserve University[1]

Future space exploration networks could leverage on high-readiness terrestrial technology. Terrestrial networks have been designed and built with a primary concern for *scalability*. This paper: Conducts an assessment of the scalability paradigm in terrestrial networks; conducts an assessment of future scalability needs for space networks, especially in the context of space exploration, and identifies gaps between space needs and terrestrial approaches. The gap analysis is then used to derive implications for potential lifecycle steps during a spiral development process.

I. Introduction

THE Internet is an extremely successful communication network, and the number of connected users and devices has been increasing exponentially [1]. The Internet has been able to accommodate a large number of hosts thanks to a fundamental design paradigm that goes under the name of scalability. *Scalability* refers to the ability of a system to sustain seamless operations when certain system parameters increase [2]. Internet scalability is a key concern that pervades all aspects of the system design. Scalability was one of the primary goals at the outset and, as Internet functionality expands, protocol designers typically consider scalability as a tacit and critical design goal.

Meanwhile, space exploration networks are characterized by a relative paucity of assets compared to terrestrial networks. As a result, space networks are not pressed for scalability, at least not in the same manner as terrestrial networks. Rather, the design of space exploration networks is dominated by considerations of communication latency, reliability, cost, and limitations on computing, power, and bandwidth. In other words, space and terrestrial networks differ at least in the relative importance of the scalability concern. The difference is significant since scalability permeates the design and implementation of the terrestrial Internet. On the other hand, the terrestrial Internet has clearly shown an impressive degree of technological maturity and cost effectiveness, and so Internet techniques have been seriously considered in the space setting [3, 4].

In this paper, we step back from specific contributions and examine the underlying scalability principle. The paper will conduct an assessment of the scalability paradigm in terrestrial network design, conducts an assessment of future scalability needs for space networks, and identifies gaps between exploration needs and terrestrial approaches. The paper will derive implications for potential lifecycle steps during a spiral development process. The discussion focuses on space exploration networks that will support the human-robotic exploration of Moon, Mars, and beyond. The paper concentrates on the network software stack, which includes layer 2 (data link) through layer 7 (application).

The paper is organized as follows. Section 2 will review the concept of scalability in terrestrial networks. Section 3 will review scalability needs in space networks. Section 4 will isolate gaps between space and terrestrial methods. Section 5 will introduce gap resolution within a spiral development process, and Section 6 will give an example. Finally, Section 7 will conclude the paper.

[1] 10900 Euclid Avenue, Cleveland, Ohio 44118 USA

II. Scalability in Terrestrial Networks

Scalability is a concept that is pervasive in terrestrial networks, and can be considered along multiple dimensions. In practice, scalability includes the four facets of numerical, geographical, administrative [5], and functional scalability.

Numerical scalability refers to the ability of a distributed system to seamlessly continue operations with an increased number of users, resources, and services. For example, a Web server should be scalable with an increasing number of clients that request documents. In the http protocol, Web servers are contacted by clients, such as browsers, which request contents. The server replies to each client individually with the requested contents. Requests and replies are processed independently, and thus each request and reply imposes a load on the server and on the network. A scalable Web server would continue to operate regardless of its load. In particular, a scalable server would operate in the presence of *flash crowds*, which are sudden spikes of client requests. Flash crowds are relatively common and can be due to, for example, a sudden interest in a piece of news on a news Web site [6]. Flash crowds and Web scalability are addressed by techniques such as Content Delivery Networks [7] and multicast [8]. More generally, Internet protocols are typically designed with particular attention to their scalability to a larger number of users, resources, and services.

Geographical scalability refers to the ability of a distributed system to support communication among users and resources that lie far apart. In particular, IP was developed with the purpose of *internetworking*, that is, of connecting any data network with each other, regardless of physical location or underlying technology. For example, an IP address in principle identifies an end-host throughout the Internet and allows connectivity between any two Internet systems. In practice, geographical scalability is typically limited to ensuring bare bone connectivity, whereby two hosts can be connected regardless of location, but no assurance is given as to the connection quality. In particular, geographically scalable systems cannot provide delay transparency due to inherent limits on the speed of light. More critically, geographical scalability does not typically include Quality-of-Service assurances, such as fault-tolerance or flow isolation.

Administrative scalability is achieved if a distributed system is easy to manage even if it encompasses multiple administrative domains. Administrative scalability is a primary goal of internetworking, in that it was felt that IP can tie together multiple existing networks only if it can respect the sense of autonomy and control of the administrators of each individual network [9]. As a result, the Internet is organized as a collection of *Autonomous Systems (AS)*, where each AS is an administration unit. For example, an AS could consist of the network operated by a large Internet Service Provider. As a result, the administration and control of the Internet is decentralized among AS's. Since Internet communication must be enabled across multiple AS's, the AS's exchange routing data through the BGP routing protocol. The BGP protocol respects and empowers each AS administrator, thus achieving administrative scalability. For example, AS *X* can specify that traffic originating from *X* must never cross AS *Y* purely on the basis of business considerations and regardless of physical connectivity or Quality-of-Service.

Numerical, geographical, and administrative scalability are well established Internet principles [2, 5]. More recent developments have isolated a fourth facet of scalability, which we designate in this paper as functional scalability. *Functional scalability* refers to the ability of a distributed system to accommodate more complex functionality. Functional scalability is a property of the algorithms and software in a distributed system. Functional scalability could be quantified in a variety of ways. For example, a distributed software system could be evaluated in terms of function points, a common measure of software complexity that is used in Software Engineering. In this metric, a distributed system becomes more complex with an increase in the number of function points. In general, and regardless of the exact metrics, Internet software is complex: for example, router software can contain more than twice as many lines of code as a core telephone switch [10]. Router complexity makes it hard to configure a network and to predict its properties once configured, and an active research area is looking at method to harness router complexity [10]. In general, Internet complexity justifies the need for functional scalability and several Internet design principles make it relatively easier to deal with various aspects of complexity. For example, the IP network layer isolates applications from the details of the underlying communication technology, so that connectivity is provided seamlessly regardless of whether the underlying network uses, say, Ethernet or Sonet. In other words, IP works as a convergence layer for multiple underlying data link protocols. It

also operates as a convergence layer for multiple applications that are written over IP regardless of which data link technology may be supported at the operational site. As a result, IP convergence greatly simplifies the effort required in writing distributed applications. More generally, the Internet increases functional scalability through a layered approach that provides for encapsulation of functionality while hiding its implementation. Similarly, distributed applications tend to share common functionality, such resource discovery or remote method invocation, and the shared method can be implemented by common middleware libraries. As a result, advanced middleware simplifies the design, development, and deployment of applications while simultaneously reducing costs and improving system quality [11]. In the space context, a middleware library could provide data marshaling for the CCSDS SLE protocol [12], as well as advanced functionality, such as remote method invocation, that are implemented in terrestrial middleware. Regardless of layers and specific issues, the emphasis on functional scalability has led to the design and implementation of expandable and reusable solutions.

Scalability is traditionally considered as the primary design objective for terrestrial networks. The reason is that terrestrial network systems have increased rapidly in terms of, for example, the number of users, the volume of data, geographic reach, the number of administrative domains, and application complexity. Furthermore, scalability is commonly considered as an assurance on quality ("if it scales, it must be working"). In general, scalability is a pervasive paradigm that must be taken into account if one wishes to exploit existing capabilities of terrestrial networks.

III. Scalability in Space Exploration Networks

Scalability seldom appears as a specific objective in the design of space exploration networks. The omission is perhaps best explained by the fact that scalability to large numbers is a fundamental facet of terrestrial scalability but significantly less relevant in space networks. Space exploration networks are rather concerned with the so-called "geographical" scalability, i.e., the potential for a distributed system to cross long communication distances. The very term, however, highlights that the original concept of "geographical" scalability was tied to terrestrial networks. Functional scalability is also important and has recently led to an emphasis on flexible, sustainable, affordable, and autonomous operations. However, space exploration traditionally ruled out certain reusable solutions, such as middleware, due to severe computational and energy limitations and long communication delays. In general, functional scalability has been sacrificed if optimized solutions are needed. As for administrative scalability, space exploration is likely to involve no more than a handful of cooperating agencies, so that the issue of scale is less pressing than in terrestrial networks. For example, an inter-domain routing protocol may not need the features of BGP that allows the protocol to scale up to tens of thousands AS's, each including its user base and its own business policies. In exploration networks, the paucity of administrative domains and of assets simplifies the design of communication protocols since large-scale administrative scalability is not an issue for the foreseeable future.

IV. Gap Analysis

The concern on scalability is ingrained in all activities of terrestrial networks. However, certain facets of scalability are irrelevant or detrimental for space applications. In the rest of this section, we will examine the four facets of scalability, discuss the commonalities between the space and terrestrial approaches, and examine the nature of technological discrepancies.

The gap is perhaps at its widest in the case of numerical scalability, since terrestrial networks can host vast numbers of communication systems, whereas space assets tend to be considerably more expensive, fewer, and sparser. In principle, numerical scalability will be critical to space network when space exploration and colonization acquires momentum, but it is far from clear that the number of space assets will significantly increase in the short or medium term. In current and planned space networks, numerical scalability is only meaningful in terms of a network that scales up as the amount of data traffic increases. In other words, the scientific community would benefit from a network that is able to convey to Earth an ever increasing deluge of data.

Another important difference is highlighted by administrative scalability. The terrestrial approach sees administrative scalability as central to network operations, since it is practically impossible to bring the

network under the control of few operators. By contrast, space networks are under the control of a single agency, or a small group of closely collaborating bodies.

Both terrestrial and space networks involve large software systems and on the surface both types of networks should be concerned with functional scalability. Terrestrial networks certainly place great attention on functional scalability. The Internet emphasizes generality and reusability, for example by means of convergence layers or middleware. Since space applications are also large, space networks also share a concern with functional scalability, especially in terms of development costs, quality, and process. However, space network are more likely to sacrifice generality and reusability in favor of reliability and optimized communication. Space networks are optimized for few expensive assets and it may make sense to go to great lengths to squeeze as much performance as possible from these assets even at the cost of customized software that exhibits relatively less generality and reusability.

Space networks are of course concerned with communication over long distances and with long latencies. Terrestrial networks are also concerned with geographic scalability over long distances and with long latencies. In particular, a point of contact may revolve around *Disruption Tolerant Networks (DTN's)* [13]. DTN's address networks characterized by long delays, predictable and unpredictable network partitions, and possibly devices with limited computational resources or power [13]. For example, a terrestrial DTN may involve a commuter bus that is equipped with a short-range wireless access point and that travels among rural villages. The users will connect to the bus as it passes along and exchange data and information with the bus. A commuter bus environment is a DTN in which messages are stored on the bus until they can be delivered to the next gateway. Such a DTN operates under network conditions that are more demanding than many space networks since rural bus schedules are typically more unpredictable than celestial dynamics and bus wireless networks are seldom operated by trained technicians. DTN's have been variously designed according to various paradigms. For example, a DTN protocol may try to hide latency and partitions from distributed applications. Another approach is exemplified by the Space Internet project and involves the creation of so-called bundles, which are relatively long messages, and the unit of communication in the DTN. For example, a DTN may include several application data units. A DTN host will store a bundle for a potentially long time until it can be forwarded to the next communication gateway. For example, a remote space asset may transfer custody of a large observation bundle to a passing satellite, which acts as a relay to deliver the observation data to Earth as soon as it comes into line of sight with a terrestrial base station.

In general, space and terrestrial network demonstrate quite different approaches to scalability and generally a cultural divide between the fields. On the other hand, terrestrial capabilities and paradigms could be useful for a successful space network.

V. Gaps and Spirals

Space and terrestrial networks show fundamental architectural differences that can be traced to their respective approach to scalability. If terrestrial paradigms are to be useful in the space context, terrestrial methods should be reconciled with space exploration requirements. This section discusses a method to integrate the benefits of the terrestrial Internet approach within a spiral development process.

In a *spiral development lifecycle*, a process set is iterated and the risk is actively managed. The spiral emphasizes risk management in that, in each cycle, the first step is to evaluate products and process, identify risk factors, and define the products of the subsequent iteration so as to minimize risk. As a result, the spiral software is effectively risk-driven in that risk analysis determines the products during each cycle. Spiral development thus differs significantly from repeatedly executing a sequence of linear (waterfall) processes blindly and with no risk assessment [14]. Specifically, the win-win spiral development consists of the following steps during each cycle:

1) Identification of the stakeholders for the next software development cycle.
2) Identification of the criteria of success of each stakeholder. In general, stakeholder will use different criteria to assess whether the proposed software system is successful.
3) Reconciliation of success criteria.
4) Risk-driven evaluation of alternatives for processes and products, identification of the objectives for the next cycle.

5) Plan execution.
6) Review and validation of product and process.

At the end of the cycle, either the product is complete or another cycle starts [14].

The spiral process can serve as a guideline to identify and resolve gaps. For example, if a terrestrial technology is proposed for use in space, the process should identify early the gaps between the technology and the space requirements, especially in terms of risk for the overall project. In this way, if a gap implies risks, it would be identified and resolved early (step 4) during each spiral cycle.

Key stakeholders are likely to bring different perspectives and success criteria to the network development process. Criteria are sometimes shared among stakeholder and sometimes different. Different success criteria would be reconciled during the first steps of each spiral cycle. Specifically, steps 1-3 involve the identifications of the system's stakeholders and their success conditions, and negotiation to determine a mutually satisfactory set of objectives, constraints, and alternatives [14]. A possible resolution is to identify additional cycles in the spiral development process. For example, the first cycle would result purely in the definition of *life-cycle objectives*, which should be accomplished after the product is fully completed. The life-cycle objectives would comprise items such as a concept of operation, the environment boundaries, prototypes of critical functions, and assurances of consistency among system elements. The next two cycles would elaborate a *life-cycle architecture*, which is the architectural design of the complete product, and comprises additional items such as an assignment of responsibilities. The life-cycle objectives and architecture function as anchor milestones whose achievement is critical to resolve key differences among stakeholders.

Spiral development breaks the development process in cycles, each of which has a shorter duration than the full project. As a result, each cycle can typically be estimated more accurately in terms of effort, size, and cost. In general, software project estimation becomes gradually more accurate as the product moves closer to completion [15], and spiral development has the effect of breaking a large project into a sequence of more manageable cycles. The spiral process has also the effect of forcing an early reconciliation between estimates and requirements. For example, a stakeholder may insist on a certain feature, another stakeholder may point out that the required feature is extremely expensive or time-consuming, and yet another stakeholder may state that she is more concerned about schedule than about a useful but non-critical feature. The spiral provides a process for the stakeholder to reconcile their differences so that requirements and estimates may be accordingly adjusted. The reconciliation should be driven by risk analysis in step 4 of the process (risk-driven evaluation of alternatives) since effort, schedule, and cost estimates are typically a major risk factor for software projects [16].

Many stakeholders have a concern about the maintainability and evolvability of any software products. For example, software upgrades may consume network bandwidth and the development of upgrades can be expensive. Furthermore, software upgrades may be needed to correct defects that were not isolated during pre-mission testing or simply to cope with changes in the mission scope. The spiral process allows stakeholder to examine maintainability and its associated risks during the first phases of each spiral cycle.

VI. Reconciliation

In this section, we conduct a thought experiment in which the early stages of a spiral cycle reconcile the success criteria of space and terrestrial networks. We emphasize that the thought experiment is purely to exemplify the process and does not necessarily reflect a recommendation on space exploration networks.

In this thought experiment, it is assumed that space communication software would place additional emphasis on generality, reusability, and maintainability, which are primary concerns of the terrestrial Internet software stack. A space network would benefit from generality and reusability since they contribute to functional scalability and cost effectiveness and maintainability. However, current space networks can sacrifice generality and reusability to optimize the communication among assets. The first spiral steps thus identify success criteria that are somewhat at odds with each other. To complicate the matter further, the proposed exploration network should interface with ground communication networks to connect provider and user sites (e.g., [12]), and the interoperability may be a pressing concern for some of the stakeholders. Finally, effort size, schedule, and cost are likely to be major concerns for most stakeholders.

The spiral process also identifies the importance of involving multiple stakeholders from the very first steps of each cycle. In particular, the spiral should identify multiple internal customers, as well as the need of involving other space agencies. Inter-agencies collaboration may be critical in future exploration missions because of the exorbitant costs of deploying space exploration assets. However, collaborations have a better chance of success if disparate stakeholders and success criteria are identified and resolved early.

A possible resolution would be to ignore generality and reusability in favor of the currently highly optimized approach. This first resolution keeps into account the extreme cost of space assets but does nothing to improve software cost or reliability The diametrically opposite resolution would be to adopt the terrestrial approach, but such an approach is likely to be tremendously expensive in space networks.

A third hybrid method would employ the terrestrial Internet paradigm as a reference model, which would be used to reason about the system and to design the distributed software. The resulting distributed application is however implemented in a form that is highly optimized for the target space application. The approach has been adopted for example to create embedded Web servers that run on microcontrollers and use no more than 1KB of source code, yet are fully compliant with the relevant Internet standards (http, TCP, IP) [17]. The Web server implementation starts with a specification that is fully compliant with the standards. In other words, the Internet standards serve as a reference model for the design of the server. The standards specify required elements, as well as recommended or optional elements. The Web server code is then reduced by stripping all of the elements that are not strictly required by the standard. The remaining software is already compact since the optional features cause most of the complexity. For example, the protocol stack needs not implement TCP congestion control or an http cache control method. At this point, the remaining Web server is subject to aggressive cross-layer optimization. In particular, the Web server implements the various http, TCP, and IP layers in one piece of code, as opposed to the standard terrestrial practice of using multiple modules. The resulting code takes about 1KB, yet it is functionally indistinguishable from a regular Web server in that messages are properly formatted in each layer and moreover the behavior of IP, TCP, and http is fully compliant with the standards. Full compliance was achieved by using the Internet standard as the starting point and as the reference model for the system design, and the compact embedded implementation was achieved by aggressively optimizing the original implementation while maintaining full standard compliance. In some sense, the approach is similar to a compiler that translates a high-level language into an optimized and machine-dependent executable.

In this approach, a benefit is that software would be designed as if it were to run on the Internet. As a result, the distributed software can leverage on the significant architectural and design effort of the terrestrial Internet. In particular, the software can be reasoned about in terms that are well grounded and the result of substantial design efforts. More generally, the software development effectively garners the benefits of generality and reusability of the terrestrial software infrastructure. Furthermore, the software is interoperable with other Internet-based systems. At the same time, the distributed system is highly optimized for use on severely constrained devices.

The approach would be even more useful if the optimization process were to be executed automatically. For example, the Web server would be implemented as in a terrestrial scenario, and then a tool would translate it into an embedded implementation by removing unused code and through aggressive cross-layer optimization. The tool is very much like an optimizing compiler that translates a high-level language into an executable. Another example is the case of software developed for embedded Java that can be optimized so as to remove all the library code that is unused in a particular application in order to reduce the code size.

Although this method achieves both generality and efficiency and it has been demonstrated in some scenarios, such as embedded Web servers, the main associated risk is that its general methodology and tools are not currently well-developed. Furthermore, the method would have to interface with terrestrial networks that are potentially more flexible, and the interface remains to be specified.

At this point, the spiral process has identified a gap and three potential resolution alternatives. The next steps are to commit to an alternative for the next cycle and to implement appropriate products and prototypes.

VII. Conclusions

Scalability is a central principle in Networks and Distributed Systems. Space networks and the terrestrial Internet differ significantly in their approach to scalability, and consequently on almost any major design choice. Although the two paradigms differ, each view can be valuable and can contribute to the development of future networks. Differences can be explicitly identified and resolved during specific steps of the spiral lifecycle process.

References

[1] Odlyzko, A. M., "The Internet and other networks: utilization rates and their implications," *Information Economics and Policy*, Vol. 12, No. 4, 2000, pp. 341-365.

[2] Tanenbaum, A. S., and van Steen, M., *Distributed Systems*, Prentice-Hall, Upper Saddle River, NJ, 2002.

[3] Liberatore, V., "Implementation Challenges in Real Time Middleware for Distributed Autonomous Systems," *Second IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT 2006)*, IEEE Computer Society, Los Alamitos, CA, 2006, pp. 21-28.

[4] "OMNI: Operating Missions as Nodes in the Internet", http://ipinspace.gsfc.nasa.gov/ [retrieved July 4, 2007].

[5] Neuman, B., "Scale in Distributed Systems," *Readings in Distributed Computing Systems*, edited by T. Casavant and M. Singhal, IEEE Computer Society, Los Alamitos, CA, pp. 463-489.

[6] Krishnamurthy, B., and Rexford, J., *Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement*, Addison-Wesley, Boston, 2001.

[7] Rabinovich, M., and Spatscheck, O., *Web Caching and Replication*, Addison-Wesley, Boston, 2002.

[8] Chrysanthis, P. K., Liberatore, V., and Pruhs, K., "Middleware Support for Multicast-based Data Dissemination: A Working Reality," *Eighth IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS 2003)*, IEEE Computer Society, Los Alamitos, CA, pp. 265-272.

[9] Cerf, V., and Kahn, R., "A protocol for packet network interconnection," *IEEE Transactions on Communications Technology*, Vol. COM-22, No. 5, pp. 627-641.

[10] Molinero-Fernández, P., McKeown, N., and Zhang, H., "Is IP going to take over the world (of communications)?" *SIGCOMM Computer Communication Review*, Vol. 33, No. 1, 2003, pp. 113-118.

[11] Mahmoud, Q., *Middleware for Communication*, Wiley, Chichester, U.K., 2004.

[12] Muzny, L., "CCSDS Space Link Extension Proposal for a NASA Wide Ground Data Service Standard", http://www.aiaa.org/Spaceops2002Archive/papers/SpaceOps02-A-T1-10.pdf [retrieved July 4, 2007].

[13] Burleigh, S., Hooke, A., Torgerson, L., Fall, K., Cerf, V., Durst, B., Scott, K., and Weiss, H., "Delay-Tolerant Networking: An Approach to Interplanetary Internet," *IEEE Communications Magazine*, Vol. 41, No. 6, 2003, pp. 128-136.

[14] Boehm, B., and Bose, P., "A Collaborative Spiral Software Process Model Based on Theory W," *International Conference on Software Process 3*, IEEE, Piscataway, NJ, 1994, pp.59-68.

[15] Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., and Selby, R., "Cost models for future software life cycle processes: COCOMO 2.0," *Annals of Software Engineering*, Vol. 1, No. 1, 1995, pp. 57-94.

[16] Jones, C., *Assessment and Control of Software Risks*, Yourdon Press, Englewood Cliffs, NJ, 1994.

[17] White, F., "webACE server", http://d116.com/ace/index.html [retrieved on July 4, 2007].