# Invocation Oriented Architecture for Agile Code and Agile Data

. Dinesh Verma*[a], Kevin Chan[b], Kin Leung[c], Athanasios Gkelias[c]
[a]IBM TJ Watson Research Center, 1110 Kitchawan Road, Yorktown Heights, NY, USA 10598,
[b]Army Research Laboratory, 2800 Powder Mill Road, Adelphi, MD, USA 20873
[c] Imperial College of London, UK 00555-9642

## ABSTRACT

In order to address the unique requirements of sensor information fusion in a tactical coalition environment, we are proposing a new architecture -- one based on the concept of invocations. An invocation is a combination of a software code and a piece of data, both managed using techniques from Information Centric networking. This paper will discuss limitations of current approaches, present the architecture for an invocation oriented architecture, illustrate how it works with an example scenario, and provide reasons for its suitability in a coalition environment.

**Keywords:** middleware, resiliency, invocations, sensor analytics

## 1. INTRODUCTION

In order to perform distributed analytics effectively, we need to explore new approaches and architectures that are suitable for different types of processing in sensor information fusion environments. Current focus on sensor information fusion has focused heavily on two types of architectures, either a streaming architecture, where sensor data is sent to a processing center using a streaming model such as Apache Edgent, or it is sent to a cloud based environment where services are implemented using a Service Oriented Architecture (SOA) or micro-services. Both of these models have limitations that make them unsuitable for a tactical coalition environment.

In a tactical coalition environment, network links are unreliable and access to services is sporadic. Access to services or systems over a network link cannot be assured. Due to dynamics of the environment, new architectures that can provide a better match for the tactical environment needs to be explored. In this paper, we propose one such architecture, an invocation oriented architecture, and examine its resiliency in a tactical environment.

In this paper, we introduce this new architecture and evaluate its suitability for distributed analytics in a tactical coalition environment. We begin with an overview of the tactical coalition environment for distributed analytics that defines the scope of our work, and then discuss how traditional stream oriented architectures and services oriented architecture models apply to this environment. We discuss the limitations of the traditional architectures in the tactical coalition environment, and introduce the new invocation oriented architecture. We present the implementation details of the architecture, discuss a couple of sample applications that can be developed using this model, and study the performance of the new architecture in the tactical environment as compared to the traditional approaches.

## 2. COALTION TACTICAL ENVIRONMENTS

We assume a coalition tactical environment as described in [1] which consists of a variety of Information Surveillance and Reconnaissance (ISR) assets that are provided by two or more coalition partners and interconnected in an ISR network. The ISR network is typically and ad-hoc network formed not just of ISR sensors [2], but can also include other sources of data, platforms, communication systems and similar devices and humans that can provide actionable Information and Intelligence (I2). The sensors can have a high degree of heterogeneity. Some sensors, like those monitoring acoustics, seismic, magnetic, passive infrared, Unattended Ground Sensors [3] and chemical/biological signals deploy persistent sensing and only send information on the network when they detect some activity. Other sensors, like day-night video, electro-optic cameras and imagers can send high resolution images in a more frequent manner. Many sensors are multi-modal, supporting more than one of the above modalities. In current and future environments, mobile ground and aerial sensing platforms will be increasingly used and will have substantial local processing capabilities and intelligence.

Within a coalition environment, assets belonging to two or more coalition operations will be used to conduct joint missions and operations. The various sensors and sensor information processing elements can be abstracted into three

types data sources, data aggregators and data sinks [4] which are interconnected in a directed graph. The data sources reflect sensors and other data generators, the data aggregators represent intermediate information fusion elements and the data sinks represent elements that process and analyze the data, either for storage, or for a human analyst to examine. The links between the various elements can be dynamic and vary in bandwidth and capacity. Because of constraints in the security policy of different coalition partners, only some data sources and aggregation functions may be visible to different members of the coalition. The setup for a coalition environment is shown in Figure 1.

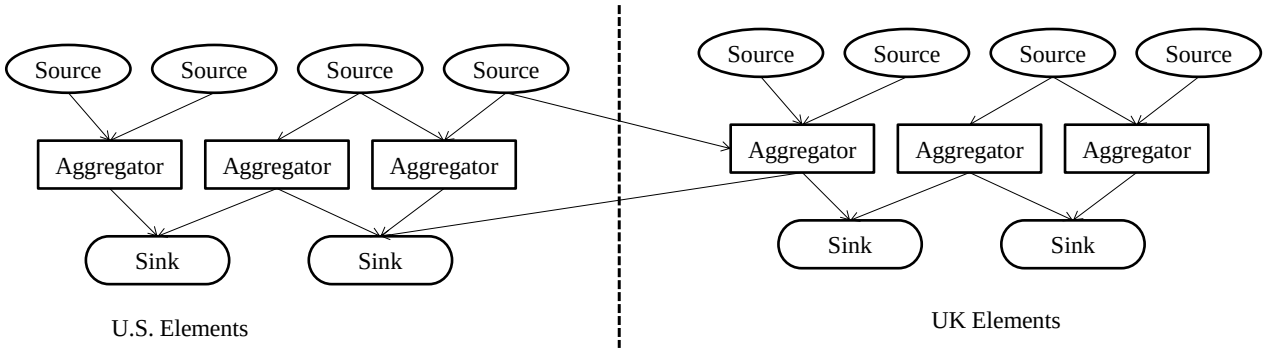


Figure 1. Information Fusion in a Coalition Environment

Each of these elements can be implemented as software modules conforming to one of the traditional architectures for sensor information processing. The most common software architectures and their suitability for the coalition tactical edge is discussed in the next section.

## 3. TRADITIONAL SENSOR ANALYTICS ARCHITECUTRES

To implement the software functions required for information processing, either a service oriented architecture or a stream processing architecture can be used. Service oriented architecture or SOA requires that each of the information processing element be implemented as a service. A service is a long running process which exports a well-known interface (API) which can be called by other software modules. Typically, the service API is called by the software module which is upstream in the directed information fusion graph as shown in Figure 1. Given proper credentials, services can be invoked across administrative domain, i.e. a U.S. service can be invoked by UK and vice-versa if proper access permissions have been provided. When network conditions are favorable and network packet loss is low, SOA services can be very effective for military operations [5].

When the services offered are lighter-weight and perform specific tasks, they are also referred to as micro-services. Conceptually, a micro-service is the same as a service, except that it is lighter-weight, may have a smaller implementation, and its API may be less complex than that of a complete service.

While the services (or micro-services) oriented architecture has been wonderful to use in the context of enterprise, cloud and backend data center environments, they are not very suitable for the coalition tactical edge. In this architecture, data sources, data aggregators and data sink algorithms will be implemented as micro-services, and these micro-services will be scattered throughout the ISR network. However, the connectivity between the different elements in the tactical battlefield environment is unreliable and frequently disrupted. This means that the invocation of a service API can be disrupted and the entire call to the service may fail abruptly. In the ISR network, ISR assets are frequently running in an unmonitored mode. If on the same device, several services are running, some of them may fail and the system may not be able to restart those services by default. Similar challenges are encountered for streaming media, where application level additional mechanisms are needed to deal with mobility and network disruption challenges [6][7].

If each asset, platform or sensor has a variety of services or micro-services, then the services need to be running in a constant manner in order for the SOA or micro-service architecture to deliver any function. Since the tactical environment has very limited systems and network management personnel capability, each such service has to be made highly resilient and reliable. Obtaining highly resilient applications is a hard problem -- since it can only be provided by means of application level modifications. As a result, a service oriented architecture in a tactical coalition environment would be either very expensive to build or have very poor resiliency. Neither of these two is a desirable outcome.

An alternative to SOA or micro-service architecture is to adopt a stream oriented architecture [8]. In the stream oriented architecture, a stream or a session of information flowing from the data source to the data sink is established. Data is processed by each of the software elements in the streaming system, and implemented to send its output to another processing unit in the stream. Stream oriented architecture can provide better scalability in many environments. However, they require that the network topology of the stream remain relatively fixed. In the tactical coalition environment, where assets are mobile and connections can be easily terminated, and nodes disappear abruptly, the assumption of a stable streaming graph cannot be satisfied.

A stream oriented architecture works best in environments where (i) the source of the information is willing to send the data (ii) there is a continuous or close to continuous amount of data to be sent and (iii) the processing required to do on the data is a long-lived compared to the time it takes to establish the stream. In a coalition environment, data sources may have restrictions on flowing across national boundaries and bandwidth is almost always limited. Similarly, there may be a lot of information fusion cases where the analytics or processing required on the data source may be short-lived, e.g. one may want to make a quick check on whether a camera has seen a specific person of interest. Due to these challenges, the stream oriented architecture is not a good match for a coalition tactical environment.

Given that the common architectures for sensor information processing are not suitable in the environment, we need to look for a new paradigm which can better address the characteristics of tactical coalition environments.

## 4. INVOCATION ORIENTED ARCHITECTURE

The invocation is the basic concept in delivering a new model for distributed analytics in a dynamic environment. An invocation is a combination of a software code and a piece of data. Conceptually, an invocation can be viewed as a single call of an API made to a service in the service oriented or micro-services architecture. However, there are some significant differences in an invocation and a service API call. An API call to a service sends some inputs to a process/service that is long running and returns an answer. In contrast, an invocation is a combination of several passive entities – a passive piece of software and one or more passive pieces of data. In an invocation, the passive piece of software becomes active momentarily, performs its operation on one or more pieces of data, and may create pieces of output data. The passivity of software distinguishes an invocation architecture from a service oriented architecture.

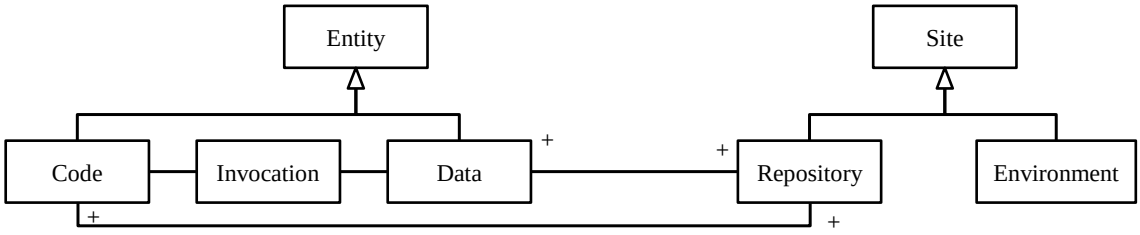


Figure 2. UML Diagram for Abstractions in Invocation Oriented Architecture

In an invocation oriented architecture, all elements (other than the implementation components described below) are passive. There are two main types of passive elements - *code* and *data*. An invocation can retrieve these passive elements from anywhere in the network, and combine them. Even when the retrieval is being made over an unreliable network, the elements can be brought in and cached locally. An invocation is live only for a relatively short time-frame -- in which the code is made active, operates on the data and then becomes passive once the operation is completed. In this model, the invocation can happen in any location within the ISR network, i.e. it is not bound to a specific site where a piece of code has been provisioned to run as a service. An invocation may take place at the location where the code is located, or it may take place in another location, e.g. where the data required is available, or at an intermediary point where both the code and the data can be brought together easily.

As passive entities, code and data elements are identified by names. Given the name of the code and the data, any location with appropriate support for invocation oriented architecture can retrieve the named code and the named data components, and combine them to complete the invocation. Such locations, where invocations can be made active, are called *sites*. Each site is this architecture can be classified as either an *environment*, or a *repository*.

An environment is a site which has the required infrastructure for invocations to be activated place. Each environment consists of a set of known services which are designed to support invocations. Since the number of such services are fixed, each of the service can be made highly reliable. A repository is a site which stores named pieces of code and data. Data sources and data sinks can be repositories. The repository supports a set of fixed services that allow the retrieval and maintenance (updates/version control) of the passive code and data available at the repository. The various abstractions in the architecture are shown by means of a UML diagram in Figure 2.

The repository and environment are both implemented as a collection of a small but fixed number of services. Because the number of services needed for supporting both types of sites is limited, each of these services can be made highly resilient. They can be designed so that they maintain heart-beats with other components, and can be restarted when down.

Each environment is required to provide an implementation of two services, an *invocation service* and a *description service.* The description service provides a way for any software module to find out the owner (e.g. the coalition partner) of the environment, and to retrieve any policies which restrict the types of invocations that will be supported at the environment, and any details of the capacity of the environment. The invocation service provides a way to give a named code and a named data (or multiple named data sets) and ask for an invocation to take place. The invocation service also provides a check interface which would tell if the invocation can be performed at the location, i.e. to ensure that it is being invoked on a compatible named data set, or that the environment has sufficient capacity to perform the invocations.

Each repository is required to provide an implementation of two services, a *retrieval service* and a *description service*. The description service provides the identity of the owner and any policies restricting the type of code or data that can be stored at the service. The retrieval service provides the CRUD interface for named code and named data, i.e. it provides an interface to create, read, update or delete any piece of named code or named data.

For easy management of the various sites, a *management service* is also running at both types of environments, which allows a central manager to look at various statistics or perform any configuration of the site.

All three services at each of the two different types of sites are implemented in a resilient manner, i.e. they maintain heart-beat messages with a recovery service, which will restart them when needed. Similarly, if the recovery service fails, one of the other services would restart it. This mechanism ensures that the basic infrastructure of each site is always available.

The system assumes that a piece of named code or named data can be retrieved by any site given its name. This requires that the underlying infrastructure support an implementation of named data networking, or otherwise, names be defined so that they encode within themselves the location from where the named entity can be retrieved. The naming of standard URIs, which encode machine names, provides a simple but useful way to satisfying this assumption.

In a tactical coalition environment, Invocation oriented architecture will be implemented by having the environment system (i.e. the three services required for the environment) active on selected nodes in the ISR network, e.g. at the UAVs, mules, or platforms, and the repository system be instantiated on sensors and at any appropriate locations in the base camp or backend infrastructure.

## 5. APPLICATION STACK FOR IOA

Invocation oriented architecture provides the middleware for writing distributed analytics applications. Writing an application on this architecture requires providing additional functionalities atop the Invocation layer. Similarly, as a middleware, invocations depend on underlying network communications layer to enable them to communicate with other nodes in the distributed analytics. The typical functions and capabilities that will be provided by the software stack that implements invocations are shown in Figure 3.

At its bottom-most layer, the application relies on a network layer which provides it the ability to transfer packets between different nodes. Such a network layer may be implemented on the usual protocols such as TCP/IP and corresponding modifications needed for mobile ad-hoc networks. An alternative would be the implementation of some disruption tolerant networking (DTN) protocols which can deal with the disruptions caused by network unreliability. Other protocols such as resilient mobile sockets might provide another possible implementation of the network layer.

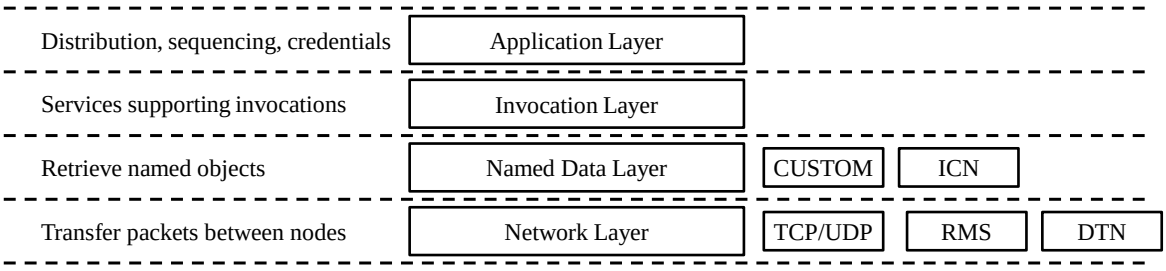


Figure 3. Layers in an application developed using invocation architecture

The named data layer provides a capability to retrieve objects which have named. The named objects for invocations could be either data objects or code. This layer may be implemented using proposed ICN approaches, or they can be coded in a custom manner with their own implementation of a method to manage named objects.

The invocation layer consists of the services described in the previous section, and consists of the services described there. The application layer consists of the user of the invocation layer, and needs to provide three key functions:

- the mechanism by which the invocation request is distributed to one or more nodes that need an invocation.
- the sequence and concatenation of different invocations
- the access control and security credentials needs to perform the invocation

Within the mobile coalition network, there are many different nods. The application needs to determine which of these nodes need specific invocations. The task of making those invocations from some location is performed by the application. Similarly, the application needs to provide those invocations with the right security credentials using which the environment can authenticate that the entity calling it is authorized to run those invocations. It is likely that an application may need not than one invocation. In that case, the specific sequence in which the invocations occur, is something that the application developer would need to specify.

In the next section, we examine two scenario applications, one in the context of coalition operations and one in a civilian context, that can be delivered effectively using the invocation oriented architecture.

## 6. SCENARIO APPLICATIONS

We consider two applications to show the value of invocation oriented architecture, a military domain one and a civilian domain one. In both domains, we compare how the application would be developed and deployed using the invocation oriented architecture versus developing it in a service oriented architecture.

### 6.1 Military Domain Application

The first application that we consider for showcasing the use of invocation oriented architecture is that of coalition operation in which a dynamic community of interest has been formed for a specific task, e.g. they are collecting video images of an area using surveillance equipment for future use. The images are stored on the local disk on a vehicle that is conducting the mission. While the operation is underway, human intelligence is received at the base indicating that there may be a camouflaged insurgent improvised explosive device (IED) along the path that they have. The mission personnel are alerted. For their safety, they would need to run and enable video analysis software on both the collected/store data as well as new data that is received so that they could identify the IED. However, the mission was launched with the goal of collecting the images and thus not prepared to do local analysis on the stored or processed video.

In this situation, invocations provide an approach by which a general mechanism on the local vehicle can augment its capabilities to perform the analysis on site. The vehicle supports the general invocation middleware, i.e. it supports the services that enable invocations to be performed locally. The base camp application would create a named code for analysis of the data providing a named source for the data that is stored at the vehicle. It would then send an invocation request to the vehicle system, which would retrieve the named code and apply it to the local data source. The sequence of operations would be as shown in Figure 4.

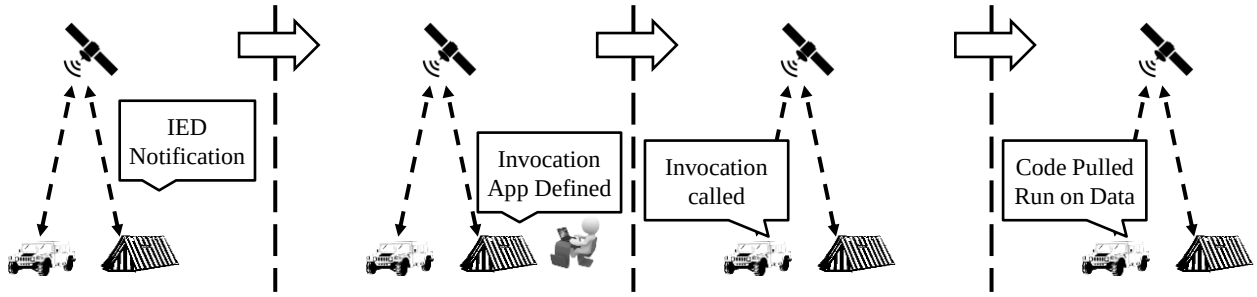


Figure 4. Sequence for calling invocations in coalition IED Scenario

In this specific situation, the invocation is a one-time call on the data that is already stored on the vehicle, and a series of periodic calls on the new data that will be collected by the vehicle. Both the stored data and the incoming data need to be assigned names -- which we assume is something an analyst at the base camp is able to do. Once the name of the newly defined invocation and the name of the data stream is sent to the vehicle, the vehicle environment pulls over the code of the invocation, runs it on the locally stored named data and the named data stream, and produced a named data set which can be stored locally. The invocation may also alert the personnel on the vehicle if the analysis found any indication of the IED in the stored data or the streaming data.

This model of invocation can be contrasted with the approach of using a SOA architecture. In the SOA architecture, a service would be created at the base-camp but it can only provide useful information if the data is sent over from the vehicle to the base-camp, which can be slow and delay-prone over the satellite link. A service on the vehicle cannot be created initially since the IED specification and details were not known at the beginning of the mission. Even if a service could be created because the vehicle personnel are computer wizards, a somewhat unlikely scenario for people whose main focus is armed combat, that service will likely not be implemented according to the best practices for resiliency and more prone to failure.

### 6.2 Civilian Domain Application

In the civilian domain, it is common to issue Amber Alerts or Silver alerts when a child or a senior is missing. The amber alert is issued on radio and displayed prominently on highway billboards for people to call in if they see a car with a specific make, model or license plate number. While the current amber alerts and silver alerts rely on humans to call into these requests, it is reasonable to expect that in future automobiles with sophisticated onboard processing capabilities, such alerts can be handled by the computers in automobiles. When an amber alert is sent out, the computer on board a car can use the cameras on the car to scan for other automobiles on the road that match the alert and can inform the central authority issuing the alert appropriately.

One challenge in automating the solution is that the authority issuing the alerts are usually local, and as such, these alerts, even when automated are not likely to conform to a uniform standard. Given that the vehicle ecosystem consists of many different manufacturers who need to operate in the jurisdiction of many different local authorities, a standardized alert system that can be supported by a common service in the vehicle is difficult to arrive at. Therefore, a single service in the vehicle that can deal with the disparate types of alerts cannot be developed and deployed.

Invocations provide a general infrastructure that can be supported in each of the vehicles. If each of the vehicle supports the software required to support an environment (as described in Section 4), then each authority can describe their own type of alerts using the invocation paradigm. The local authority defines the invocation which includes a description of the type of analytics that needs to be performed. The only standardization that is needed is that a common naming approach for the output of the video from the camera mounted on the vehicle be followed. Even if that is not standardized, an invocation can easily map it to the right name used from a table mapping the name for the video stream using the make and model of the automobile. This allows the local authority to send the specifics of their amber alert to all the vehicles that are on the road currently.

Upon receiving the request to call the invocation, the vehicles pull in the required named code from the local authority site, and apply the analytics to the feed from their onboard camera. When a match is obtained, the invoked application can call another invocation to notify the vehicle being searched for at the environment of the local authority. The sequence of calling invocations in this scenario is shown in Figure 5.

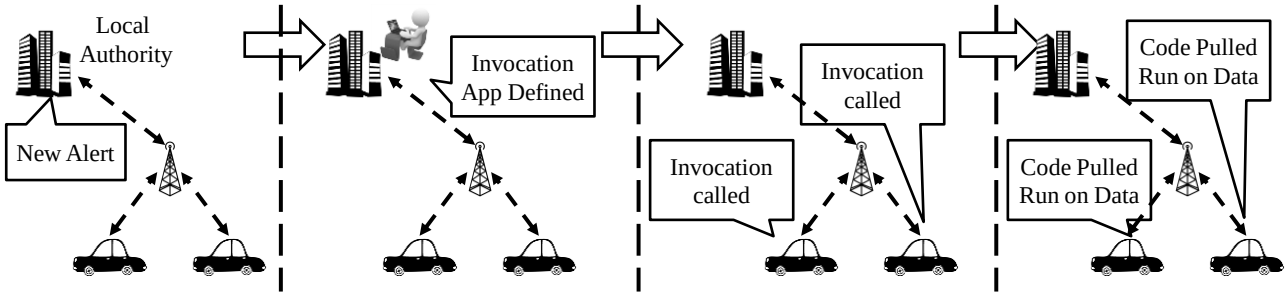


Figure 5. Sequence for calling invocations in civilian alert scenario

## 7. PERFORMANCE ANALYSIS

We compare the relative resiliency and latency of running an application using the invocation oriented architecture and the traditional service oriented architecture in a tactical coalition environment. While we are not explicitly comparing the performance of the stream oriented paradigm, the resiliency and latency of the two traditional paradigms, service oriented architecture and stream oriented architecture will remain the same, the difference primarily being whether s streaming protocol is used to access functions running at a node in the network, or whether a request-response type of protocol is used for the same.

In a tactical coalition network, the assets borrowed from different coalition partners are usually linked together in an ad-hoc network, which is mobile and the links as well as the nodes are not as reliable. This network could be supported by a backend infrastructure. Let us consider a scenario where a node in the mobile ad-hoc network has some data, and is able to perform some analytics on it by means of invoking an analytics application. Any distributed analytics application which requires K supporting services (or micro-services) can then be implemented in one of the following four manners:

i. The solution is implemented as a set of *K* services running in the backend. The data node accesses the first one of these services over the mobile ad-hoc network, which then calls the services over the fixed infrastructure in the backend. Data is transmitted over the mobile ad-hoc network to the service backend.

ii. The solution is implemented as a set of K services running on nodes in the backend. The data node sends the data to the appropriate service for the right calls to be made.

iii. The solution is implemented as a set of K services running on the data node.

iv. The solution is implemented as a set of invocation calls on the data node.

The four possible choices are shown in Figure 6. The first model in the figure is that of running the analytics in the backend as a collection of several services. This approach is shown as (a) Backend SOA. In Backend SOA, the data needs to be transferred to the first service in the backend. After that multiple calls are made to the services running in the backend. The second model of (b) Network SOA has the services running in the backend network. This model avoids the delay in reaching the backend service, but has to encounter the lossy nature of the links in the ad-hoc network at the tactical edge. The third model is to run all the services at the data node, show as (c) Local SOA in the figure. The final option is to run a series of invocations at the data node shown as (d) IOA. Each of these models would have different resiliency and latency properties.

In order to understand the resiliency, let us consider the resiliency of the services that are implemented in each of the different models. Services implemented in the backend can be assumed to have a significant level of resiliency, just as the limited number of services used to implement the invocation oriented architecture. The resiliency of network implemented services on nodes in the mobile ad-hoc network is likely to be somewhat lower. The resiliency of implementing all the services on a single node is likely to be similarly lower, since these services cannot be provided the same level of support that a backend system can be provided. Network SOA also has to deal with the unreliability of the link that connects the different services.

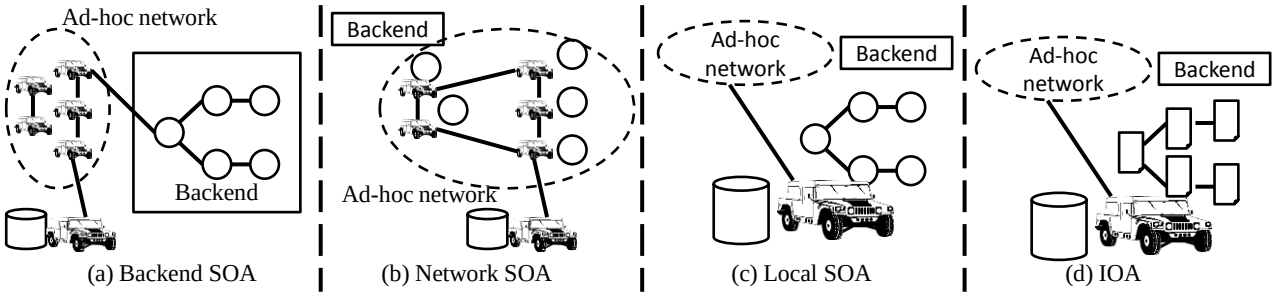


Figure 6. The different models for analytics in a tactical coalition environment

Given the resiliency of different services and the network link, we can determine the resiliency of the overall approach for all four models of the solutions. Assuming that services and network links fail independently, the resiliency of the overall service can be computed as the probability that all the services are up. We can then plot the resiliency of the different approaches as a function of the number of services that need to be invoked.

Figure 7 shows the plot of service resiliency against the number of component services with the following metric values:

| **Metric** | **Value** | **Justification** |
|---|---|---|
| Backend Service Resiliency | 0.999 | Backend services can be made highly reliable, and 3 9s is fairly common |
| IoA Support Service Resiliency | 0.999 | IoA services are limited in number, so can be made highly reliable |
| IoA Support Service Count | 4 | As per discussion in Section 4. |
| Network Service Resiliency | 0.99 | Network services on nodes in tactical field is less reliable than backend |
| Local Service Resiliency | 0.9 | Local service needs to be installed ad-hoc, and thus is least reliable |
| Network Link Resiliency | 0.8 | Network links in ad-hoc networks are not very reliable. |

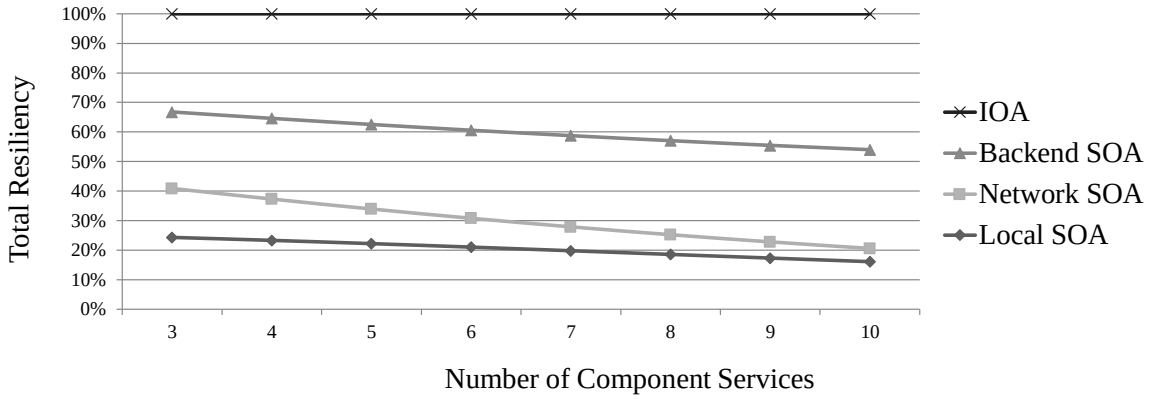


Figure 7. Resiliency of different analytics models with increasing number of component services

As shown in Figure 7, the invocation oriented architecture maintains the highest level of resiliency, since its component services are not likely to fail. Backend SOA provides the next level of resiliency, but suffers from the high loss nature of the ad-hoc network. The local SOA model does not perform very well due to the lower resiliency of each of the constituent services and the network SOA has the lowest resiliency since the unreliable ad-hoc network needs to be traversed multiple times.

Even if we assume that a general local service can be implemented as reliably as a network service, the resiliency of local SOA solution is not as good as that of the IOA, which relies on a fixed set of services. Figure 8 shows the

resiliency of overall solution when we assume that local service resiliency is same as network service resiliency, namely 0.99. While the local SOA approach improves in resiliency under this assumption, it still is highly unreliable compared to IOA.

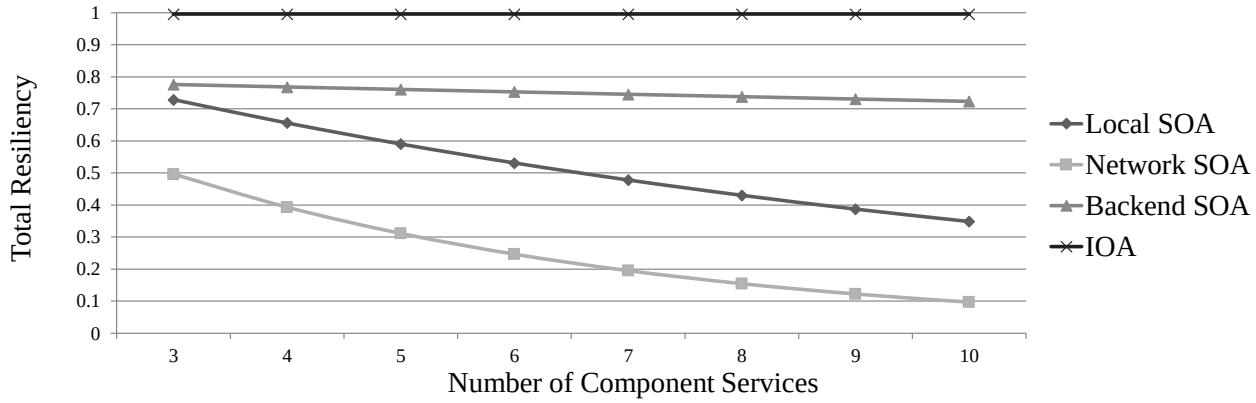


Figure 8. Resiliency of different analytics models when local services are assumed to be more resilient

Another benefit that the IOA offers over network SOA or backend SOA architecture is that of reduced latency in completing an operation when the amount of data to be transferred is less than the amount of code to be retrieved to conduct the operation. This benefit will be realized by both a local SOA implementation as well as an IOA implementation since they both rely on local processing alone. Figure 9 shows the net latency of the network SOA and backend SOA latency in processing an analytics function when the data required for analytics is 5 times the size of the processing code, traversing over an ad-hoc network with links of 80% reliability. The numbers are scaled so that the IOA latency is a unit, and provides a scale for comparison. In this case, the latency for backend SOA as well as network SOA are several times larger than the IOA approach.

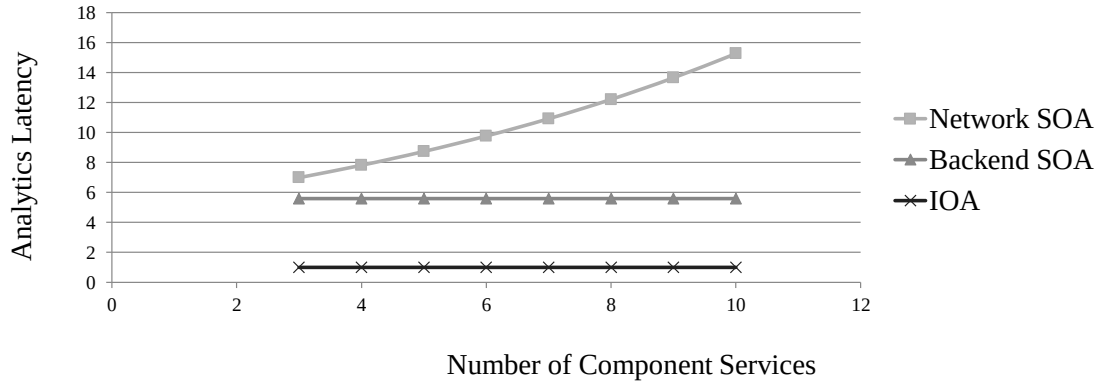


Figure 9. Comparison of operation completion latency in different architectures

Figure 10 shows the latency incurred as a function of the reliability of the network link. As expected, while the backend SOA latency increases with the link, the network SOA model is the one whose latency increases rapidly as the network reliability decreases.

We can use the same analysis to determine the resiliency and latency for the applications in the two scenarios discussed in the previous section. In the military scenario, a lot of data is stored on the vehicle, and the resiliency of the service installed in the ad-hoc manner is not going to be high. Thus, for both qualitative reasons as well as quantitative measurements, IoA would provide the best solution for that case. For the civilian alert scenario as well, the service

installed locally with be very error-prone, and transferring video feeds from several cars to a service will not be efficient. IoA would provide the best solution model in that case as well.

In military networks, nodes usually have limited storage and memory, but can be expected to be relatively homogenous. The nature of analytics tends to be highly variable depending on the mission scope and task. On the other hand, civilian devices have larger storage, and the nature of analytics in any specific domain (e.g. public safety) may be less diverse than in military networks, although there are many more civilian domains for which analytics may be needed. Despite these fundamental differences, IoA approach can address the limitations arising in both types of networks effectively.

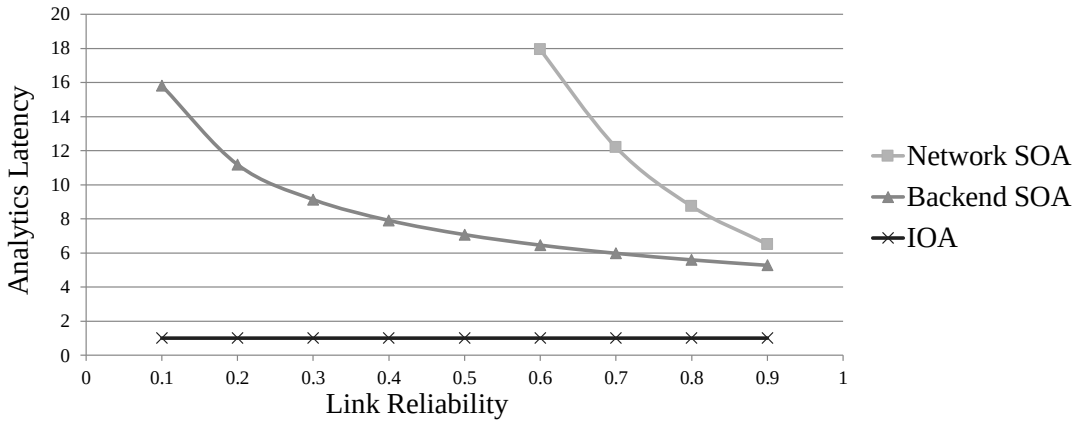


Figure 10. Latency incurred by network and backend SOA models compared to IOA as a function of network link reliability

## 8. ACKNOWLEDGEMENTS

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copy-right notation hereon.

## REFERENCES

[1] Pham, T., Cirincione, G. H., Verma, D., & Pearson, G., "Intelligence, surveillance, and reconnaissance fusion for coalition operations," Proc. IEEE 11th International Conference on Information Fusion, 1-8 (2008).

[2] Fraden, J., [Handbook of Modern Sensors], Springer-Verlag, New York (2004).

[3] Stotts, L.B., "Unattended-ground-sensor-related technologies: an Army perspective", Proc. SPIE 4040, AeroSense, 1-9 (2002).

[4] Aberer, K., Hauswirth, M., & Salehi, A., "Infrastructure for data processing in large-scale interconnected sensor networks," Proc. IEEE International Conference on Mobile Data Management, 198-205 (2007).

[5] Manso, M., Calero, J.M.A., Barz, C., Bloebaum, T.H., Chan, K., Jansen, N., Johnsen, F.T., Markarian, G., Meiler, P.P., Owens, I. and Sliwa, J., "SOA and Wireless Mobile Networks in the tactical domain: Results from experiments," Proc. IEEE Military Communications Conference-MILCOM, 593-598 (2015).

[6] Kristiansson, J., and Parnes, P., "Application-layer mobility support for streaming real-time media," Proc. IEEE Wireless Communications and Networking Conference, 268-273 (2004).

[7] Gao, W., and Cao, G., "User-centric data dissemination in disruption tolerant networks," Proc. IEEE INFOCOM, 3119-3127 (2011).

[8] Cherniack, M., Balakrishnan, H., Balazinska, M., Carney, D., Cetintemel, U., Xing, Y., and Zdonik, S. B., "Scalable Distributed Stream Processing," Proc. First Biennial Conference on Innovative Data Systems Research, 257-268 (2003).