Opportunistic Named Functions in Disruption-tolerant Emergency Networks

Pablo Graubner
University of Marburg, Germany
graubner@informatik.uni-marburg.de

Patrick Lampe
University of Marburg &
TU Darmstadt, Germany
lampep@informatik.uni-marburg.de

Jonas Höchst
University of Marburg &
TU Darmstadt, Germany
hoechst@informatik.uni-marburg.de

Lars Baumgärtner
University of Marburg, Germany
lbaumgaertner@informatik.uni-marburg.de

Mira Mezini
TU Darmstadt, Germany
mezini@informatik.tu-darmstadt.de

Bernd Freisleben
University of Marburg &
TU Darmstadt, Germany
freisleb@informatik.uni-marburg.de

ABSTRACT

Information-centric disruption-tolerant networks (ICN-DTNs) are useful to re-establish mobile communication in disaster scenarios when telecommunication infrastructures are partially or completely unavailable. In this paper, we present opportunistic named functions, a novel approach to operate ICN-DTNs during emergencies. Affected people and first responders use their mobile devices to specify their interests in particular content and/or application-specific functions that are then executed in the network on the fly, either partially or totally, in an opportunistic manner. Opportunistic named functions rely on user-defined interests and on locally optimal decisions based on battery lifetimes and device capabilities. In the presented emergency scenario, they are used to preprocess, analyze, integrate and transfer information extracted from images produced by smartphone cameras, with the aim of supporting the search for missing persons and the assessment of critical conditions in a disaster area. Experimental results show that opportunistic named functions reduce network congestion and improve battery lifetime in a network of battery-powered sensors, mobile devices, and mobile routers, while delivering crucial information to carry out situation analysis in disasters.

ACM Reference Format:
Pablo Graubner, Patrick Lampe, Jonas Höchst, Lars Baumgärtner, Mira Mezini, and Bernd Freisleben. 2018. Opportunistic Named Functions in Disruption-tolerant Emergency Networks. In *CF '18: CF '18: Computing Frontiers Conference, May 8–10, 2018, Ischia, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3203217.3203234

1 INTRODUCTION

Facilities like power stations, water reservoirs, and telecommunication centers are part of the critical infrastructures that are vital for modern societies. During and in the aftermath of a disaster or an emergency event, such as an earthquake or a terrorist attack, it is essential to maintain these infrastructures and to restore and to repair capacities that have been damaged. Telephone lines, cellular base stations and parts of the Internet backbone might be destroyed, disrupted or overloaded due to network congestion. However, information about the current situation is crucial for affected people and rescue teams. Thus, it is important to re-establish basic means of communication during a disaster despite fragmented IP networks and totally or temporarily disrupted network links.

In the past, mobile ad-hoc networks (MANETs) and disruption-tolerant networks (DTNs) were studied as approaches to re-establish basic communication services during disasters. More recently, information-centric network (ICN) protocols for disaster scenarios, such as ICN-MANETs [18] and ICN-DTNs [2, 5, 16] were proposed. Apart from important challenges such as authentication and access control, ICN protocols address the following aspects in a disaster scenario: (i) since end-to-end connectivity is not guaranteed and location-based as well as fixed addresses may not work, name resolution at the network layer (instead of at the application-layer) can be used, which also supports anchor-less mobility, (ii) since caching, traffic engineering and prioritization based on the name of the desired content is an inherent concept of ICNs, network nodes can make a trade-off between, for example, bandwidth and storage based on the relevance of the content for a consumer. Furthermore, Named Function Networking (NFN) [25] as a generalization of Named Data Networking (NDN) [11] offers great potential for providing support in disaster scenarios. In NFN, names do not only refer to data, but also to functions and computational tasks, and the network's role is to resolve names to computations.

In this paper, we present *Opportunistic Named Functions* (ONFs), a novel approach to extend the NFN paradigm for DTNs in disaster scenarios. ONFs are *opportunistic* in the sense that (i) named functions are applied based on locally optimal decisions using criteria such as network congestion avoidance, battery lifetime and device capabilities, and (ii) named functions are applied after the receipt of data transmitted during opportunistic communication. In particular, the paper makes the following contributions: (1) We present a novel approach for ICN-DTNs in which ONFs are used for data preprocessing, analysis, integration and transfer within wireless networks in disaster scenarios where in-network processing can provide essential information for situation analysis. (2) We introduce a novel implementation of ONFs within Serval, a well evaluated open-source DTN project for disaster situations [4].

CF '18, May 8–10, 2018, Ischia, Italy

ACM ISBN 978-1-4503-5761-6/18/05…$15.00
https://doi.org/10.1145/3203217.3203234

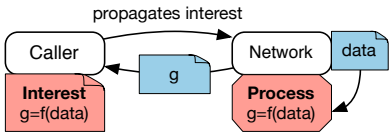


Figure 1: Basic ONF concept

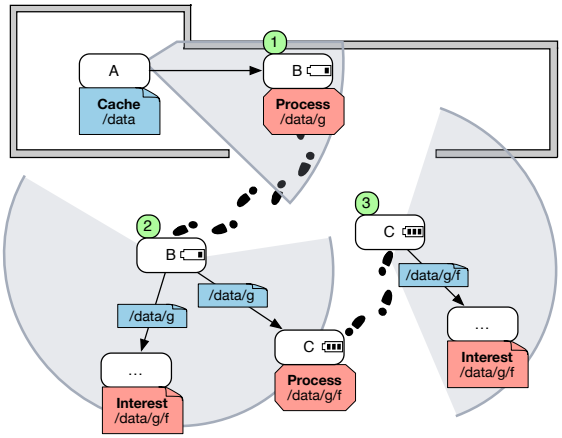


Figure 2: Example of in-network processing of named content with ONFs

(3) We propose novel methods for image (pre-)processing and face detection in disaster scenarios to support the search for missing persons. We further present experimental evaluations with respect to battery consumption and runtime of these methods. (4) We present simulations for using ONFs in basic network topologies and a disaster scenario.

The paper is organized as follows. Section 2 presents the design of ONFs. Section 3 describes our scenario, the search for missing persons in a disaster. Section 4 describes the implementation of ONFs. Our implementation is evaluated in Section 5. Section 6 discusses related work. Section 7 concludes the paper and outlines areas of future work.

2 OPPORTUNISTIC NAMED FUNCTIONS

In Named Function Networks (NFNs) [21, 24, 25], the network orchestrates function execution and caching, in an intelligent manner. For resource-constrained ICN-DTNs, the collaboration between nodes to efficiently execute functions and cache intermediate results is also highly desirable, but opportunistic communication introduces several problems: (i) communication is driven by content consumers *and* by intermittent opportunities, (ii) routing paths, computing nodes, and caches might be (temporarily) unavailable, (iii) global information about the network, i.e., its topology, statistics about different link qualities, and other information relevant for performing an orchestration is incomplete or sparse in the best case and unavailable in the worst case, (iv) mobile devices and sensors are resource-constrained and have limited battery capacity, and (v) these devices do not have a common architecture and might not be capable of performing complex functions.

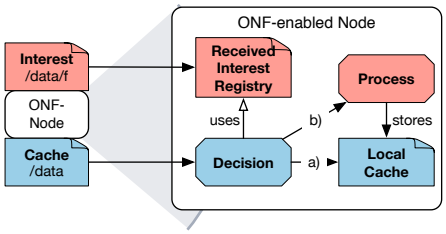


Figure 3: Functionality of ONFs

To address these problems, we propose *opportunistic* named functions (ONFs). Figure 1 shows the basic idea of our approach. A function $f(data)$ is defined to be processed in the network. The node specifies a request to the network, similar to an asynchronous function call, where the node is the caller and the network is the callee. Each data producer in the network can either (i) perform the requested function on the original data or (ii) delegate the execution of the function to other nodes. In the best case, the request is processed completely within the network and the results are returned to the callee. In the worst case, the network delegates function execution on the produced data back to the callee, where $f(data)$ is then applied.

Figure 2 illustrates in-network processing with ONFs in an ICN-DTN. Node A (upper left) stores cached content `/data`, which can serve as input for two kinds of function: `g(/data)` and `f(g(/data))`. In our approach, functions applied on existing content are specified using a naming scheme, where `g(/data)` is represented by `/data/g`. Furthermore, nodes signal their interest in the result of a function by declaring interest packets. In Figure 2, green points (1-3) describe different points in time. At point (1), node B is in the range of node A that opportunistically transmits the cached content `/data` to node B. Node B, on the other hand, is aware of the two interests `/data/g` and `/data/g/f`, respectively, and after the reception of the data, node B decides how to process the content based on the known interests. In our example, the remaining battery of the node is low, so only one of two possible functions is applied and the result is stored in its cache. Then, node B moves to point (2), where it is in the range of more nodes, including node C. Since DTNs are based on the store-and-forward principle, all nodes in range receive the cached content. Since node C has a full battery, C decides to process `/data/g/f` and store the result in its cache. When node C arrives at point (3), the results of the function is transmitted to another node. To summarize, the data traverses the network from the content producer to the interested consumer, while it is processed on intermediate nodes.

Figure 3 shows the architecture of our ONF approach. First, a content consumer specifies his/her interest in content. This is expressed as an interest packet transmitted to an ONF-enabled next hop and then propagated opportunistically in the ICN-DTN network. An interest is specified as a human-readable hierarchical name that may refer to both content or functions. During interest propagation, at each hop, the received interests are registered locally. When new content is received from another node, the registered interests in specific content are used to execute opportunistic named functions. ONFs are applied based on locally optimal criteria as well as the

priority of the registered interest. Afterwards, the results are cached locally. If a node has an intermittent connection to one or more other nodes, it sends the cached content in the order of its priority to the other nodes. This is explained in more detail below.

2.1 Interest Propagation and Registration

In existing ICN approaches (e.g., CCNx), a Forwarding Information Base (FIB) at each hop specifies the path on which an interest packet will be forwarded. Data packets traverse the network on the backroute of an received interest packet, stored in the Pending Interest Tables (PITs) at each hop. In ICN-DTN, the propagation of a content request is not coupled to forwarding/routing mechanisms. Decoupling function execution and routing allows us to use different DTN routing algorithms. In our approach, interest and data packets travel on independent paths through the network. We use a bundle protocol on top of epidemic routing, detailed in Section 4.

Due to the loose coupling between content request resolution and routing/forwarding, a received interest might be outdated or the content might have already been delivered by another node. Received interests are stored in a Received Interest Registry (RIR). When a content consumer is no longer interested in specific content, a negative interest packet is sent by the client to the next ONF-enabled hop to propagate through the network similar to an interest packet. The RIR is also used for basic bookkeeping purposes. It stores the last received interest or negative interest per user, and decides whether a received interest is newer than a stored interest. Therefore, an interest/negative interest packet needs a globally unique identification for the user and a sequence number.

2.2 Hierarchical Naming

Interests in ONFs are specified as human-readable hierarchical names that may refer to both content or functions, such as `/camera/persons/injured`. In this example, an operation is specified on the topic `/camera`, which represents all camera images taken from all participants. When a picture is taken with a mobile phone's camera, a hook in the camera software library throws an event received by the ONF library. The ONF library then calls the name resolution engine described in Section 2.4, which is then responsible for performing the operation `persons/injured` on the camera image. Since ICN-DTNs do not use a global registry for specific files/filenames, our design of the hierarchical naming scheme does not allow us to request a specific file like `/camera/image_001.jpg`. Instead of specifying a file request to the network, the content consumer specifies a topic and receives the corresponding content.

In addition to topics and topologies, the hierarchical naming scheme is used to express a *pipeline* of functions. For example, a content consumer can express his/her interest in all pictures taken at a specific location in the form of a filter on GPS latitude and longitude and a filter on all images containing a person `/camera/GPS_coordinates_50_8/persons`. Here, latitude and longitude are parameters for the filter function `GPS_coordinates`, encoded in the hierarchical name. When a content producer receives interests, then the node, based on the local context and its capabilities, decides which kind of and how many functions it applies on the produced content. It might store pictures under the name `/camera` and deliver them opportunistically to other nodes, where the GPS function is applied with the specified parameters. If the GPS filter is applied, the content is named `/camera/GPS_coordinates_50_8` afterwards. It either continues to detect `persons`, or it transmits the resulting image to another node able to perform the detection.

In case of multiple interests specified by one or more content consumers, the hierarchical naming is used to resolve *common tasks* that might be useful to serve as preliminary results for multiple interests *at once*. For example, in a disaster scenario, medical workers might be interested in all pictures taken from injured persons, specified by interest `/camera/persons/injured`. Rescue teams might be interested in crowds of people, specified by `/camera/persons/crowd`, to organize supply or transportation for them. In these cases, the common interest in `/camera/persons` might be used for additional savings: either saving CPU cycles by forwarding the content of `/camera/persons` without detecting injuries or a crowd to both nodes, or saving transmission time by forwarding only pictures containing injuries or a crowd.

2.3 Alternative Names

ONFs rely on hierarchical and alternative names. While hierarchical names can be interpreted as subsequent executions (similar to an `AND` operator in several computer languages), alternative names introduce alternative function executions (similar to an `XOR` operator in several computer languages). This is especially interesting in scenarios where a slight reduction of the quality of results might be acceptable. For example, if a content consumer is interested in all images of fires specified by `/camera/fire`, it might not be possible to deliver a picture taken with a 16 megapixel camera to the content consumer, since interest- and packet-routes in an ICN-DTN are inherently non-deterministic and without guarantees. Instead, it is more useful to specify an alternative to that interest, with a higher possibility of a successful network traversal. For example, a user can specify the alternative `/camera/fire or /camera/wavelet_filter`, where wavelet transformations can be used for fire detection [23] that can run efficiently on digital signal processors, in contrast to a memory-intensive visual concept detection. In this example, alternative names are used to prioritize the interests with an `XOR` operator: if it is not possible to deliver results by visual concept detection, the network should use a wavelet transformation as a fallback. This concept is not limited to a single alternative. If multiple alternatives are specified, they are interpreted as an (ordered) cascade of fallbacks.

2.4 Name Resolution and Function Execution

Algorithm 1 is used to resolve names in ONFs. First, when new content is received, the device's context is used to get a list of functions that are available and applicable based on a device's context. Then, a loop computes the next functions to be applied on the content. This is used to allow pipelined execution of functions. Inside the loop, the resolvable interests are computed based on their naming scheme. For example, an interest in `/camera/fire` is applicable to the name `/camera`. Then, the corresponding functions are parsed using the longest prefix-match method that allows unambiguous name resolution.

The most important part is to decide which functions should be applied. This can be seen in the two functions `getPriorityMatches`

Algorithm 1: Name Resolution

Input: *name*: received content name, F: set of available functions, I: registered interests

Function *parseFunction(interests, functions)*
  for *i in interests* **do**
    i.function ← longestPrefixMatch(*functions*);
  return *interests*.functions;
Function *getPriorityMatches(parsedFunctions, functions)*
  for *f in parsedFunctions* **do**
    matches ← priority of f in *functions*;
  return *matches*;
Function *getBestMatch(interests, matches)*
  result = ∅;
  sorted_matches = sort *matches*: $m_1 > m_2$ if m_1 has more interests than m_2;
  for *m in sorted sorted_matches* **do**
    result ← *result* ∪ *m.function*;
    if *all interests applied* **then**
      break;
  return *result*;
F' ← get applicable functions from F;
for *not F'.isEmpty()* **do**
  I' ← get resolvable interests for *name* in I;
  functions ← parseFunction(I', F');
  matches ← getPriorityMatches(*functions*, F');
  M ← getBestMatch(*matches*);
  name ← apply each function in M on *name*;
  F' ← get applicable functions from F;

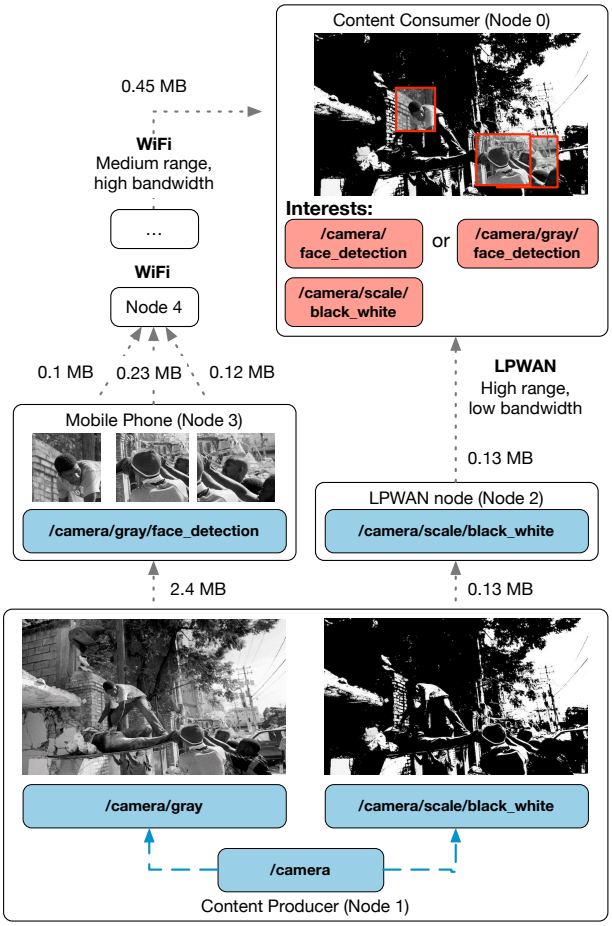


Figure 4: Processing ONFs in a disaster scenario

and `getBestMatch`. The latter returns the functions that should be applied at this stage of the pipeline. For example, if both interests `/camera/persons/injured` and `/camera/persons/crowd` are specified, both functions are applied to the content with the name `/camera`. If there is an interest in `/camera`, the identity function is returned. More importantly, the function `getPriorityMatches` computes a total order of all functions, which is used to decide which functions should be applied. Therefore, each alternative name in an interest is handled in the order of its occurrence; the first name is handled with priority 1, the second with priority 2 etc. Then, the number of interests that can be served by a function according to its priority is used to compare each function, i.e., a function that serves two interests with priority 1 is preferred to another function that serves two interests with a lower priority. After the functions are executed, the loop is restarted at the next stage.

3 OPPORTUNISTIC NAMED FUNCTIONS IN DISASTER SCENARIOS

In a disaster scenario, the search for missing persons is a primary task for rescue teams and highly important for affected people. In such situations, sensors, battery-powered mobile devices, and generator-powered routers remaining from the communication infrastructure can be used to establish a resilient disaster-response communication system. The inherent tradeoffs between either spending CPU cycles and battery power for ONF function execution, or spending battery power and air time for data transmissions, is illustrated by the following example. A content consumer is interested in all detected faces in all pictures taken by the mobile devices' cameras. Since state-of-the-art mobile phone cameras usually provide a resolution of 8-16 megapixels, simply transferring all raw images from the content producer to the consumer has the risk of producing network congestion. In contrast, ONFs can be used to (i) delegate function execution, (ii) apply multiple stages of image preprocessing that can be deployed even on small devices and can be executed in a pipeline, and (iii) prioritize processed content based on the importance for a content consumer.

In our scenario, ONFs are used to support the search for missing people in image files that are distributed via an ICN-DTN. Figure 4 illustrates the processing of ONFs in an ICN-DTN within this disaster scenario. The first node at the top of the picture is a command center or the head of a rescue team that relies on information about missing persons. This is reflected by the specified interests in content, which in this case are also formulated with domain knowledge

about the applicable filters, scalings, and lossy compressions. In a real-world scenario, this process could be transparently handled by an additional application provided to rescue teams. In our scenario, we have built a small software component that relies on detected faces, either in their original color or transformed to a grayscale image, and low-resolution black/white images. The component visualizes the information by showing detected faces in high resolution, with a low-resolution black/white-underlay for situation analysis. It is not sufficient to transfer the low-resolution part only, since it is not possible to perform face detection algorithms on black/white images with accurate results [13].

In Figure 4, we assume that all nodes already received the interest packets sent by the consumer. The producer at the bottom of the figure is a microcontroller-based camera that is able to make basic image transformations, such as grayscaling and shrinking a black/white version of an image. In our case, the resulting images are transferred to two different nodes: an ARM-based smartphone that can perform a face detection algorithm and a microcontroller-based Low Power Wide Area Network (LPWAN) device. Due to its low bandwidth, the microcontroller-based LPWAN device cannot transfer a 2.4 MB image with 25 kbps in reasonable time. On the other hand, the mobile phone can perform a face detection on the grayscaled image. It extracts 3 faces (0.1 MB, 0.23 MB and 0.12 MB) that are then transmitted via 802.11 WiFi. At the content consumer, the scaled black/white image and each of the three detected faces arrive separately. The software component on top of the ICN-DTN is notified at each arrival and provides an image to the user that is successively completed with a low-resolution black/white image in the background and grayscale faces in the foreground, which can be used for situation-awareness. To summarize, this scenario illustrates the operation of ONFs in ICN-DTNs (i) without end-to-end connectivity (ii) with unpredictable routes, and (iii) with heterogenous devices on the route through the network.

4 IMPLEMENTATION

In our disaster scenario, ONFs are used to support the search for missing persons through image files that are distributed via an ICN-DTN. Therefore, ONFs have been integrated into the Serval Project [7–9] that is centered around a suite of protocols designed to allow infrastructure-independent communication based on DTN.

The implementation of the ONF functionality in Serval, as well as the applied functions, are described below. Image transformation and face detection functions are introduced that can be used to trade between execution time, resource consumption and quality of results.

4.1 Serval-based ONFs

The Serval Project provides software for mobile devices to form a secure, self-organizing and fully distributed mesh network. Serval's store-and-forward DTN protocol (Rhizome) allows network operation in the absence of end-to-end connectivity and can run on top of a transport-agnostic Mesh Datagram Protocol (MDP). MDP can run both (i) on top of the IP protocol stack or (ii) on bare link layer protocols like packet radio. Serval Rhizome implements a simple stateless flooding protocol running in user space. Since no guarantees for meeting other nodes are given, epidemic flooding of content to neighbors is used, providing fault tolerance and reliability at the expense of consuming resources. Data transmission is based on broadcast (announcements) and unicast (packet transfer), and the data can be transferred un- or encrypted and/or signed. These user space network layer mechanisms run with acceptable performance, both on common MIPS-processor based access point/router hardware and on low-end ARM smartphones [7].

To provide basic ICN capabilities, we have designed an interface for specifying interests. Interests are specified in the form `/camera/face_detection`. Similar to other ICN-DTN instances [16], our implementation distinguishes between interest packets and content packets. They are realized with interest and content *bundles* in Serval Rhizome, which are transferred to other nodes without internal fragmentation or scattering. Furthermore, we have integrated generic hooks for handling incoming and outgoing payload by independent user space programs into Serval, called Serval Hooks [1]. These hooks allow user space applications to control (i) announcements of existing data to other nodes to forward interest and content bundles to other nodes at the sender, (ii) filtering of bundles at the receiver based on bundle metadata, and (iii) processing of received bundles. Consuming and producing bundles is handled by the same mechanism, i.e., a node can act as a content producer, an ONF node, or a content consumer at the same time.

An ONF-enabled Serval node maintains the RIR, a name resolution component, and a function execution environment. Functions are implemented as executable binaries that reside either (i) in a designated folder inside the local file system, or (ii) as data bundles within the DTN network, allowing the distribution of (signed and encrypted) binaries via the network. When a new content bundle arrives at an ONF-enabled node, the name resolution algorithm decides on the basis of the content's name whether it is cached as it is or a function is applied (or both). In many scenarios, a pre-installed set of functions is available, such as official mobile warning apps distributed by governments and installed by mobile users.

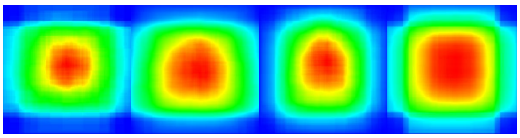
4.2 Implemented ONFs

The implemented ONFs for our scenario are described below.

4.2.1 Image (Pre-)Processing. For basic image preprocessing purposes, we use the OpenCV[2] library that is available for multiple hardware architectures and software platforms. For black/white conversion of images, we use a basic thresholding operation to filter the intensity of each pixel above a threshold and assign to it a black value or a white value, respectively. Thus, we convert the original image to a 1-bit black-and-white image. To achieve grayscaling, we use an 8-bit grayscale operation integrated into OpenCV. Image scaling is performed by a resampling method using pixel area relations, which reduces the effect of moiré patterns in contrast to interpolation methods.

4.2.2 Smart Image Fragmentation. Another approach to trade result quality for transmission time is smart image fragmentation. It consists of two independent steps: (i) to fragment an image, where each fragment represents a region of the original image, and (ii) to

[1] source code available at https://github.com/umr-ds/serval-dna/tree/nicer-hooks
[2] http://opencv.org



(a) Fire (b) Helmet (c) Gas Mask (d) Building

Figure 5: Regions of interest for relevant topics

prioritize and filter the fragments statistically, according to their relevance to detect a face or a visual concept.

This approach is related to the field of detection proposals [10] for machine learning approaches and makes use of the property that bounding boxes of interest are usually found within a specific region, and most importantly close to image centers [14]. Figure 5 illustrates this property by showing regions of high interest for relevant topics with red color and regions of low interest with blue color in a heatmap. Therefore, heatmaps were generated from bounding boxes provided by the ImageNet[3] image library that organizes images according to the WordNet[4] hierarchy, i.e., according to meaningful concepts, described by multiple words or word phrases. For each topic, a matrix with 256×256 values was generated. These matrices represent the number of overlaps between the bounding boxes for a specific region of a normalized image. All bounding boxes for all detected objects of a topic were retrieved from the database and were scaled according to a normalized image size.

When image fragments are filtered according to these matrices, their relevance is determined *statistically* and not by inspecting a specific image. Assuming that fragmentation is performed during the compression phase and the matrices representing the statistical relevance are stored in memory, then this image fragmentation technique has $O(1)$ runtime for a single memory lookup. Thus, smart image fragmentation can be applied on devices with small amounts of memory (KBs instead of MBs) and constrained computation capabilities. It possibly sacrifices some result quality for a reduced amount of data that needs to be transmitted.

4.2.3 Face Detection. For face detection, we apply a two-stage approach. In the first stage, the Viola/Jones algorithm [26] implemented in *OpenCV* is used, since it is a common out-of-the-box solution for face detection. It is comparatively fast [6] and has a low false negative rate. Thus, as many faces as possible are detected quickly. However, it has a relatively high false positive rate that is acceptable if the algorithm is used as a pre-filter. In the second stage, *dlib*[5] is used to verify the results. It is slower than the Viola/Jones algorithm, but in the second stage it operates only on small subimages that can be processed much faster. Additonally, *dlib* has a low false positive rate and a higher precision [6]. Consequently, *dlib* can act as a validator for the results produced by the Viola/Jones algorithm. All parameters except the face sizes are independent of the used face detection algorithms. Thus, the algorithms can be replaced by alternatives, still benefiting from the rest of our optimizations.

[3]http://image-net.org
[4]http://wordnet.princeton.edu/
[5]http://dlib.net

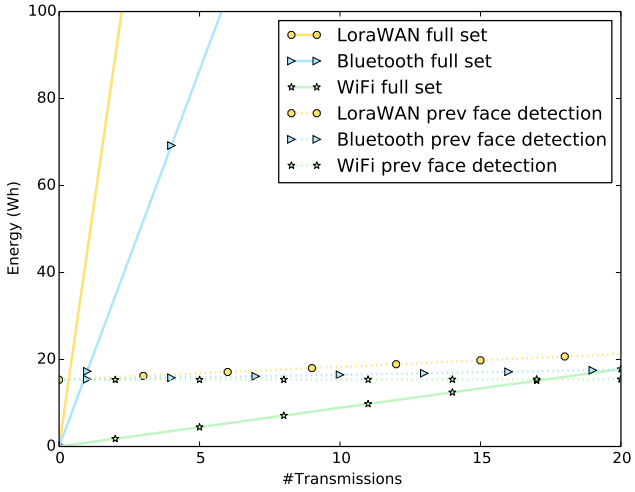


Figure 6: Energy consumed for a transmission with and without a previously applied face detection

5 EXPERIMENTAL EVALUATION

We present an experimental evaluation of ONFs below. In Section 5.1, image (pre-)processing, smart image fragmentation, and face detection are evaluated in terms of runtime and power consumption. In Section 5.2, these experimental results are used to simulate ONFs in different basic topologies as well as in a disaster scenario.

5.1 ONF Measurements

To evaluate image (pre-)processing, smart image fragmentation, and face detection as examples of ONFs, we performed measurements on a Raspberry Pi 3, model B (RPi). This device is used as a reference for several devices with ARM-based CPUs, such as mobile phones. As a test image set, we used an existing image set that only contains images related to emergency scenarios [13]. It consists of 1,482 files, with a total size of 2.7 GB. It includes the following scenario specific search terms on an Internet image search engine: *Haiti earthquake, earthquake faces, earthquake people, disaster people, disaster faces.* Power consumption was measured using the ODROID Smart Power meter with a sampling frequency of 5 Hz.

The experimental results show that black/white transformations of all images consume a total of 0.35 Wh (328 seconds at an average of 3.8 W), grayscaling consumes 0.31 Wh (296 sec at avg. 3.86 W), image resizing consumes 1.01 Wh (964 sec at avg. 3.79W), and face detection consumes 15.33 Wh (14,914 sec at avg. 3.7 W). While the average power consumption differs only slightly during this test, it shows that each face detection takes, on the average, 10.1 sec, with a standard deviation of 4.7 sec. Smart image fragmentation has been tested with pictures fragmented into matrices of 128×128 images. In our tests, we reduced the total amount of transmitted data at the following rates: 5%, 10% and 25%. A face detection applied afterwards was also possible, but the reduction of 10% gave the most reliable results of 94.6% positive detections.

To estimate a possible tradeoff between the transfer of an image and an expensive computation like a face detection, we measured

the power consumption of a LoRa interface (connected to a RPi), a Bluetooth interface (via two RPis) and a WiFi interface (on the RPi during a transmission of our test data set). We then compared these measurements of a data transfer with energy consumed when (i) a face detection was performed and only smaller images within bounding boxes of the detected faces were transmitted, and (ii) this smaller data set was transferred via different wireless links. Figure 6 shows both the energy consumed for the full data set as well as the data set resulting from a previous face detection. The x-axis shows the number of hops while the images traverse the network, the y-axis shows the energy consumption over time. Since the resulting data set is quite small compared to the original data (only 18 MB or 0.75%), the energy spent for transmission on links with lower bandwidth (LoRaWAN: 25 kbps, Bluetooth: 600 kbps) dominates the total energy consumption. While for these links a face detection before the first transmission is useful, the break-even point for WiFi is not reached before 16 hops. Applying an optimal strategy would require global knowledge about the network topology and its constituent links, which cannot be assumed in a network relying on opportunistic communication.

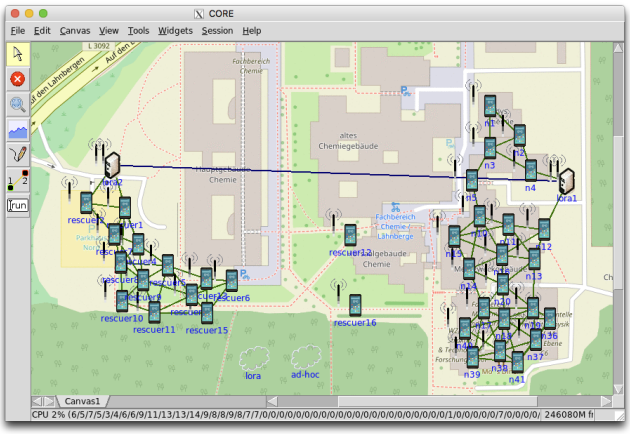


Figure 7: Disaster scenario modeled in CORE. Rescuers approaching from the left, trapped people on the right.

5.2 Network Topologies

We used the experimental results from the previous section to simulate ONFs in different basic topologies as well as in a disaster scenario. The described topologies were implemented using the CORE (Common Open Research Emulator) network emulation framework. CORE provides WiFi access point and ad-hoc characteristics, traffic shaping, as well as simple network bridges for direct linking of certain nodes [1].

5.2.1 Topologies. We investigated three topologies: *Chain*, *Star*, and *Disaster*. The *Chain* topology consists of 64 mobile phone nodes that are connected pairwise in a chain. In this topology, the first node in the chain is defined as the content producer and the last one as the consumer. The *Star* topology consists of one middle node that acts as a gateway to all other nodes. The middle node is evaluated in two different configurations: (i) a lightweight router, only featuring basic data processing functions, and (ii) a powerful wireless router that is also capable of more complex functions like face detection. Figure 7 shows a screenshot of *Disaster* scenario, a topology as it could be found in an emergency event. In the disaster site, 22 people are trapped building a partial mesh and 15 rescuers are moving in the direction of the trapped people also building a partial mesh. When the rescuers approach the building the two mesh networks merge. To support the approaching rescuers, the trapped people document different aspects of the disaster using their mobile phones. A LoRa link is established to enable basic text communication and to propagate the interests of the rescuers.

5.2.2 Simulation Results. Based on these topologies, we performed several simulation experiments. The results are presented in Table 1. The *{Chain, Star, Disaster} raw* scenarios do not execute any functions, instead the images are spread to all nodes. They provide a baseline for comparison to *{Chain, Star, Disaster} with ONFs*. While the *Chain* scenario is evaluated with one content consumer and one producer, the *Star* scenario is evaluated in two different configurations: in the *Star-get* variants, all except one node act as content producers, whereas in the *Star-share* variant only one node acts as a producer and all other nodes act as content consumers. The *Star* scenarios are further evaluated in two different configurations. In the lightweight configuration, the middle node acts as a simple router only providing inexpensive functions, such as smart image fragmentation. In the second variant, the router is also capable of applying the functions mobile phones provide, such as face detection. Finally, the ONF scenarios use our interest propagation and function application mechanism. These scenarios can be used to investigate different metrics applicable to the proposed mechanism (e.g., interest propagation time) as well as general metrics for comparison with the other scenarios (e.g., content delivery time). For the ONF scenarios, the interests are specified similar to our scenario in Section 3: a first interest specifies a face detection on `/camera`, alternatively on fragmented or grayscaled images. A second interest specifies an image scaling performed on the original images or on a grayscaled- or a black/white-version of the image.

Note that the decision whether an ONF is executed or not depends on the devices' context. In our implementation, the expensive face detection functions are skipped if the battery level of a device is lower than 50%. Instead, grayscaling, fragmentation and black-/white transformations are applied. The results of our simulations are shown in Table 1. The value t_c specifies the content transmission time, while the value t_i specifies the interest propagation time. Furthermore, the total amount of transmitted data and the estimated energy consumed are listed. The latter is modeled by (i) E_{trans}, the energy consumed during transferring data at 10 Mbps via WiFi (at 1.72 W, according to our experiments with the Raspberry Pi) and (ii) E_{comp}, the total amount of energy consumed by the applied ONF, according to the Raspberry Pi measurements.

Table 1 shows significant differences between *raw* and the ONF-based scenarios, although the same number of images were stored in `/camera` by the content producers. For the *Disaster* scenario, although no ONFs are performed and no faces are detected, the transmission energy spent by 1.5 GB of data is significantly higher (36%) than with ONFs. This is due to the data reduction of both

Table 1: Scenario tests

Name	avg(t_c)	max(t_c)	avg(t_i)	$\sum$ bytes	E_{trans}	E_{comp}	E_{sum}
Disaster with ONFs	19.41 s	214.22 s	3.68 s	371.2 MB	510.50 Ws	836.2 Ws	1346.72 Ws
Disaster raw	50.85 s	268.97 s	-	1.5 GB	2119.04 Ws	-	2119.04 Ws
Chain with ONFs	27.42 s	101.08 s	7.76 s	488.4 MB	671.49 Ws	597.3 Ws	1268.79 Ws
Chain raw	79.56 s	389.77 s	-	2.8 GB	3881.70 Ws	-	3881.70 Ws
Star-get with ONF	8.33 s	20.11 s	0.27 s	131.3 MB	180.26 Ws	836.2 Ws	1016.48 Ws
Star-get raw	15.32 s	40.83 s	-	366.2 MB	503.62 Ws	-	503.62 Ws
Star-share with ONF	10.20 s	27.81 s	2.08 s	131.3 MB	180.26 Ws	836.2 Ws	1016.48 Ws
Star-share with ONF (lightw.)	11.01 s	31.04 s	4.08 s	354.8 MB	487.10 Ws	21.0 Ws	508.10 Ws
Star-share with ONF (router)	20.71 s	44.85 s	5.17 s	94.7 MB	129.34 Ws	784.5 Ws	913.90 Ws
Star-share raw	19.06 s	45.43 s	-	412.0 MB	566.91 Ws	-	566.91 Ws

scaling and face detection on the devices. Although more transmissions were performed (1475 compared to 702) due to multiple interests, the propagation time of the interest was also relatively low (3.68 sec), since the small interest bundles could be transmitted via the wide-range and low-bandwidth LoRaWAN. Using ONFs, the average content transfer time could be reduced by more than 60%.

In the *Chain* scenario, significantly more energy is consumed without ONFs (factor 3). In this long chain of nodes between consumer and producer, the latter scales images and detects faces according to the specified interest. Due to the high number of hops between consumer and producer, the interest propagation time t_i is rather long and the total amount of transmitted data is very high (488.4 MB vs. 2.8 GB). Therefore, the total energy consumption within this network is dominated by data transmissions. The content propagation time t_c is very high for the raw case and can be reduced to by again more than 65% on the average and almost 75% in the maximum case.

In the *Star* scenario, the interest and content propagation times are short due to the small number of hops between producer and consumer. *Star-get with ONF* and *Star-share with ONF* share the same behavior, since all ONF functions are executed on the producer nodes. Due to the small number of hops, the energy for function execution dominates the total energy consumed by this network. In such a case, ONFs deliver function results, but cannot utilize ONF for better energy efficiency. In contrast, a lightweight router in scenario *Star-get with ONF (lightweight)*, only featuring basic data processing functions is able to perform inexpensive tasks to reduce the total amount of data (14%) and the consumed energy (10.5%). In the more powerful router in scenario *Star-get with ONF (router)*, a face detection function is applied at the router. This is less energy-consuming compared to ONFs at the producer (10%), but it also cannot utilize ONFs for better energy efficiency. In the first two *Star-share* scenarios, content transmission times were reduced by a small amount, showing again a small advantage of the approach.

To summarize, the basic topologies show that ONFs applied at the producer are quite advantageous in scenarios with a high number of hops between consumer and producer, but in topologies with smaller numbers of hops, ONFs can be utilized for network congestion avoidance, but not for higher energy efficiency. We have shown that, even though ONFs require a rather high amount of time for computation, ONFs lead to shorter content delivery times. In a disaster scenario, on the other hand, a significant amount of traffic (factor 4) and energy (36%) can be saved compared to not using ONFs and flooding the network with raw images.

6 RELATED WORK

ICN-DTNs were recently used for message-based communication in disaster scenario notifications [5, 12, 19]. Here, naming schemes were utilized to specify a group of receivers (e.g., the police) and to prioritize the importance of certain messages (e.g., a SOS message is more important than a chat message). Within our ONF approach, roles and multiple receivers can also be expressed with the help of hierarchical names, i.e., specific function executions are only delivered to certain roles.

Monticelli et al. [16] assume that during a disaster, the network in densely populated areas is fragmented into smaller community networks. The authors leverage vehicles or mobile phone users wandering around to exchange ICN packets between isolated network fragments. In contrast to such mobile phone-only scenarios, radio technologies (e.g., Bluetooth, LoRaWAN, WiFi, TETRA digital radio or satellite links) introduce more complex topologies of heterogenous connections (between multiple classes of devices), which can be utilized by our ONF proposal with respect to transferring a smaller amount of data via high-range links with at a low data rate.

The typical approach to perform face detection on mobile devices is to offload images to a server and run a face detection algorithm [22]. Although generator-powered servers might also be available in a disaster scenario, rescue teams and affected people cannot rely on it. In the domain of DTNs for disaster communication, preprocessing and delivery of medical images for healthcare workers were applied by Ashar et al. [3] and Roy et al. [20]. In contrast to their application-specific implementation, ONFs can be leveraged to allow more applications running in parallel, and possibly benefit from each other by sharing a common set of preprocessed data.

Named Function Networking (NFN) was proposed by Tschudin et al. [25]. It was realized based on Content-Centric Networking (CCN) and used, e.g., as a method for realizing Content Delivery Networks (CDNs). In these approaches, partial or complete function execution

is coupled with the ICN forwarding mechanism. In contrast, in ONF networks, routing and forwarding are decoupled from function execution. This allows functions to be applied opportunistically, hence we rely on *opportunistic* named functions (ONFs).

Melvix et al. [15] proposed a context- and tolerance-based forwarding strategy for IoT and 5G scenarios. Tolerances are used to tolerate longer delays, precomputed or approximate data instead of real-time sensor values. NFN is optionally leveraged to perform arithmetic functions on the data received from sensors. This is similar to our proposal of alternative names, but in contrast to our approach, it is specific to sensor data processing and aggregation.

Nguyen et al. [17] presented a directional interest propagation mechanism for crowd sensing in NDNs. This approach distributes interest packets by only re-broadcasting them if the receiver is nearer to the area of interest then the last sender. Traveling interest packets are minimized and the network load is reduced. In contrast, we distribute the interest packets to all participants and reduce the data size of the results by applying our ONFs.

7 CONCLUSION

In this paper, we have presented opportunistic named functions as a novel approach to operate ICN-DTNs during emergencies. Affected people and first responders use their mobile devices to specify their interests in particular content and/or application-specific functions that are then executed in the network on the fly, either partially or totally, in an opportunistic manner. Opportunistic named functions rely on user-defined interests as well as on locally optimal decisions using criteria such as battery lifetimes and device capabilities. In the presented emergency scenario, they were used to preprocess, analyze, integrate and transfer information extracted from images produced by their smartphone cameras, with the aim of supporting the search for missing persons and the assessment of critical conditions in a disaster area. Experimental results have shown that opportunistic named functions reduce network congestion and improve battery lifetimes in a heterogeneous network of battery-powered sensors, mobile devices, and mobile routers, while delivering crucial information to carry out situation analysis in a disaster. Using ONFs in our disaster scenario saves a significant amount of traffic (factor 4) and energy (36%) compared to not using ONFs by flooding the network with raw images.

There are several areas for future work, such as (i) exploring incentive mechanisms for function execution to perform functions locally for a global benefit, possibly by leveraging a game-theoretic approach, and (ii) providing further applications for delay tolerant networks, such as remote medical support in regions with sparse populations as well as during a disaster.

ACKNOWLEDGMENT

This work is funded by the LOEWE initiative (Hessen, Germany) within the NICER project and the Deutsche Forschungsgemeinschaft (DFG, SFB 1053 - MAKI).

REFERENCES

[1] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim. CORE: A Real-time Network Emulator. In *Military Communications Conference*, pages 1–7. IEEE, 2008.

[2] C. Anastasiades, T. Schmid, J. Weber, and T. Braun. Information-centric Content Retrieval for Delay-tolerant Networks. *Computer Networks*, 107:194 – 207, 2016.

[3] M. Ashar, H. Suwa, Y. Arakawa, and K. Yasumoto. Priority Medical Image Delivery Using DTN for Healthcare Workers in Volcanic Emergency. *Scientific Phone Apps and Mobile Devices*, 2(1):9, 2016.

[4] L. Baumgärtner, P. Gardner-Stephen, P. Graubner, J. Lakeman, J. Höchst, P. Lampe, N. Schmidt, S. Schulz, A. Sterz, and B. Freisleben. An experimental evaluation of delay-tolerant networking with serval. In *Global Humanitarian Technology Conference (GHTC), 2016*, pages 70–79. IEEE, 2016.

[5] J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. CNS: Content-oriented Notification Service for Managing Disasters. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ICN '16, pages 122–131, 2016.

[6] J. Cheney, B. Klein, A. K. Jain, and B. F. Klare. Unconstrained Face Detection: State of the Art Baseline and Challenges. In *International Conference on Biometrics (ICB '15)*, pages 229–236. IEEE, 2015.

[7] P. Gardner-Stephen, A. Bettison, R. Challans, and J. Lakeman. The Rational Behind The Serval Network Layer For Resilient Communications. *Journal of Computer Science*, 9(12):1680, 2013.

[8] P. Gardner-Stephen, R. Challans, J. Lakeman, A. Bettison, D. Gardner-Stephen, and M. Lloyd. The Serval Mesh: A Platform for Resilient Communications in Disaster & Crisis. In *IEEE Global Humanitarian Technology Conference (GHTC)*, pages 162–166. IEEE, 2013.

[9] P. Gardner-Stephen, J. Lakeman, R. Challans, C. Wallis, A. Stulman, and Y. Haddad. MeshMS: Ad Hoc Data Transfer within a Mesh Network. *International Journal of Communications, Network and System Sciences*, 8(5):496–504, 2012.

[10] J. Hosang, R. Benenson, P. Dollar, and B. Schiele. What Makes for Effective Detection Proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):814–830, 2016.

[11] V. Jacobson, D. K. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking Named Content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pages 1–12. ACM, 2009.

[12] S. Kim, Y. Urata, Y. Koizumi, and T. Hasegawa. Power-saving NDN-based Message Delivery based on Collaborative Communication in Disasters. In *The 21st IEEE International Workshop on Local and Metropolitan Area Networks*, 2015.

[13] P. Lampe, L. Baumgärtner, R. Steinmetz, and B. Freisleben. SmartFace: Efficient Face Detection on Smartphones for Wireless On-demand Emergency Networks. In *24th Int. Conf. on Telecommunications (ICT 2017)*, Limassol, Cyprus, 2017.

[14] V. Mahadevan and N. Vasconcelos. Biologically Inspired Object Tracking Using Center-Surround Saliency Mechanisms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(3):541–554, 2013.

[15] L. Melvix J.S.M., V. Lokesh, and G. C. Polyzos. Energy Efficient Context Based Forwarding Strategy in Named Data Networking of Things. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ICN '16, 2015.

[16] E. Monticelli, B. M. Schubert, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. An Information Centric Approach for Communications in Disaster Situations. In *IEEE 20th Int. Workshop on Local Metropolitan Area Networks*, pages 1–6, 2014.

[17] T. A. B. Nguyen, P. Agnihotri, C. Meurisch, M. Luthra, R. Dwarakanath, J. Blendin, D. Böhnstedt, M. Zink, and R. Steinmetz. Efficient Crowd Sensing Task Distribution Through Context-aware NDN-based Geocast. In *42nd IEEE Conference on Local Computer Networks (LCN'17)*, pages 52–60. IEEE, 2017.

[18] S. Y. Oh, D. Lau, and M. Gerla. Content Centric Networking in Tactical and Emergency MANETs. In *2010 IFIP Wireless Days*, pages 1–5, 2010.

[19] I. Psaras, L. Saino, M. Arumaithurai, K. K. Ramakrishnan, and G. Pavlou. Name-based Replication Priorities in Disaster Cases. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 434–439, 2014.

[20] A. Roy, S. Mahanta, M. Tripathy, S. Ghosh, and S. Bal. Health Condition Identification of Affected People in Post Disaster Area Using DTN. In *IEEE 7th Ann. Ubiquitous Computing, Electronics Mobile Communication Conf. (UEMCON)*, pages 1–3, 2016.

[21] M. Sifalakis, B. Kohler, C. Scherb, and C. Tschudin. An Information Centric Network for Computing the Distribution of Computations. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ICN '14, pages 137–146, 2014.

[22] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman. Cloud-vision: Real-time Face Recognition Using a Mobile-Cloudlet-Cloud Acceleration Architecture. In *IEEE Symposium on Computers and Communications (ISCC 2012)*, pages 59–66. IEEE, 2012.

[23] B. U. Töreyin, Y. Dedeoglu, U. Güdübay, and A. E. Cetin. Computer Vision Based Method for Real-time Fire and Flame Detection. *Pattern Recognition Letters*, 27(1):49 – 58, 2006.

[24] C. Tschudin and M. Sifalakis. Named Functions for Media Delivery Orchestration. In *20th International Packet Video Workshop 2013*, 2013.

[25] C. Tschudin and M. Sifalakis. Named Functions and Cached Computations. In *IEEE 11th Consumer Communications and Networking Conference (CCNC)*, pages 851–857, 2014.

[26] P. Viola and M. J. Jones. Robust Real-time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.