Aggregate Programming for the Internet of Things:

Aggregate programming provides an alternative that simplifies the design and creation of complex iot systems, it is useful for large-scale scenarios (crowds), in a physical environment with integrated devices. It addresses problems following 3 principles:

1. the “machine” being programmed is a region of the computational environment whose specific details are abstracted away.
2. the program is specified as manipulation of data constructs with spatial and temporal extent across that region.
3. these manipulations are executed by the individual devices in the region.

Advantages: Take advantage of interactions between smartphones to estimate crowd density/distribution.

A device-centric programmer must focus on protocol and how interactions will produce a desired behavior.
An aggregate programmer reasons from services and continuous type data structures, generating a data structure that maps from location to crowd density.

Field calculus: interaction patterns, captures essential features in a universal language for mathematical analysis. The unifying abstraction of field calculus is a field, which is built using four program constructs: Built in functions, dynamics, interaction, restrictions.

Building blocks for resilient coordination: adds resiliency, identifies collection of general basic operators for coordination applications, consists of self-stabilizing coordination mechanisms, 3 generalized coordination operators: G(source, init, metric, accumulate): generalizes the measurement of distance, transmission and projection, C(potential, accumulate, local, null): accumulates information in the source, T(initial, floor, decay): Flexible countdown with variable rate in time.

The developed libraries are then applied and combined with operators to form pragmatic APIs, and the application code is written.

Hybrid models are needed that take advantage of the capabilities of aggregated programming and cloud-based architectures. Security must also be considered, as IoT environments are open and dynamic.

Architectures for distributed and hierarchical Model Predictive Control

Economic and technological reasons have motivated the development of processes, but these can be difficult to control with centralized control structures due to their complexity, due to this, distributed control structures have been developed. (Methods based on Model Predictive Control (MPC)).

MPC can guarantee system stability against disturbances, or against uncertain models and transforms the control problem into an optimization problem.

Decentralized control:
Most large-scale industrial systems have decentralized structures, where their control and controlled variables are separate sets. Very few decentralized MPC algorithms with guaranteed properties have been developed so far, reasons: for large-scale systems it may be convenient to decompose the general optimization problem into several smaller problems, the MPC law is implicit and control variables are the solution of an optimization procedure instead of being computed by an explicit control law.

Distributed control:
When local regulators are designed with MPC, the transmitted information consists of locally computed future predicted control variables, so that any local regulator can predict the interaction effects. Depending on
The communication network: fully connected algorithms or partially connected algorithms.
Exchange of information between local regulators: non-iterative algorithms or iterative algorithms.
General iterative procedure: independent algorithms or cooperative algorithms.

Hierarchical control for coordination:
The basic idea is to describe the global system where the inputs of one subsystem are the outputs or states of another.
For any subsystem, an optimization problem is solved with MPC by minimizing an appropriate local cost function. If the calculated local solutions have consistency between the values of the calculated variables, the procedure ends. Otherwise, an iterative "price coordination" method is used. These optimal prices are sent to low-level local optimizers that recalculate the optimal paths of the variables.
The iterations stop when the interconnected variables satisfy the required consistency conditions.

Hierarchical control of multilayer systems:
It has: Hierarchical Control of Multi-Time Scale Systems (slow speeds and fast dynamics), Control of Hierarchical Structured Systems (The highest stratum of the hierarchy corresponds to a slow dynamic system. middle layer of hierarchy corresponds to subsystems that need to be controlled at a higher rate) and Hierarchical Control for Whole Plant Optimization (In the upper layer, Real Time Optimization (RTO) is performed to calculate the optimal operating conditions).

Coordinated control of independent systems:
It is related to the problem of coordinating a series of agents that must cooperate to globally minimize a cost function subject to joint restrictions. Instead of solving a single centralized control problem, it is possible to solve a series of local optimization problems and coordinate the local actions of the agents through an adequate exchange of information.

Related Design Problems, Open Topics, and Conclusions:
There are still many problems to solve:
New algorithms with guaranteed properties, Control structure selection, Reconfigurable control structures and hybrid systems, Optimization algorithms, Distributed state estimation, System partitioning, Synchronization and communication protocols.